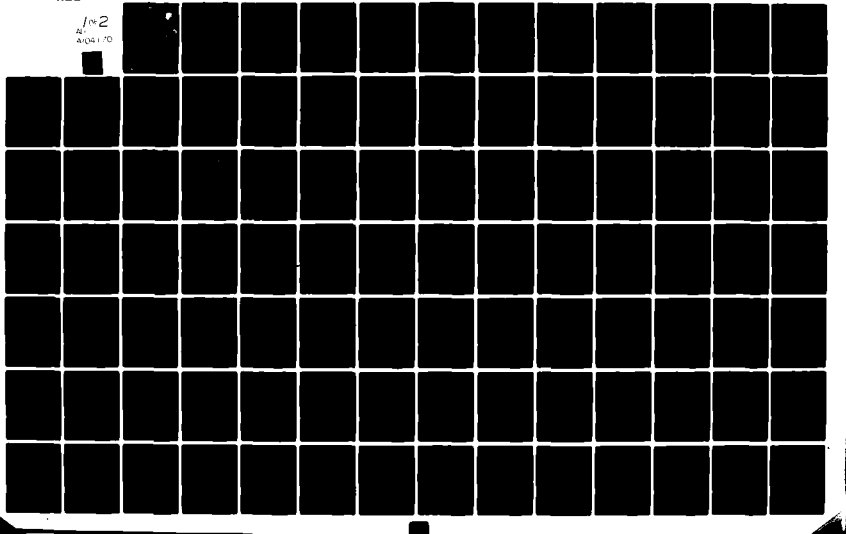
AD-A104 170 GENERAL ELECTRIC CO PITTSFIELD MA ORDNANCE SYSTEMS F/G 9/2
ELECTRICAL CHARACTERIZATION OF ADVANCED MICROPROCESSORS,(U)
JUN 81 B W HAJDUK, T M OSTROWSKI, B NEWTON F30602-80-C-0041
UNCLASSIFIED RADC-TR-81-126 NL



LEVEL II

RADC-TR-81-126
Final Technical Report
June 1981

ELECTRICAL CHARACTERIZATION OF ADVANCED MICROPROCESSORS

General Electric Company

Barney Hajduk, et al

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-126 has been reviewed and is approved for publication.

APPROVED: Regis C. Hilow

REGIS C. HILOW
Project Engineer

APPROVED: David C. Luke

DAVID C. LUKE, Colonel, USAF
Chief, Reliability & Compatibility Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBRA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-126	2. GOVT ACCESSION NO. AD-A104 170	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ELECTRICAL CHARACTERIZATION OF ADVANCED MICROPROCESSORS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report June 1981 6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Barney W. Hajduk, et al		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS General Electric Company/Ordnance Systems 100 Plastics Ave Pittsfield MA 01201		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 23380187
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RBRA) Griffiss AFB NY 13441		12. REPORT DATE June 1981 13. NUMBER OF PAGES 138
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Regis C. Hilow (RBRA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprocessor
Electrical Testing
Benchmark
MIL-M-38510 Slash Sheet

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The objective of this effort was to develop functional and parametric tests for selected microprocessors and to develop MIL-M-38510 slash sheets for them. A test pattern and program were developed for the Z8001 and data was taken and analyzed to determine its operating region. A general benchmark was also developed and used to compare the performance of the 8086 and Z8000.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

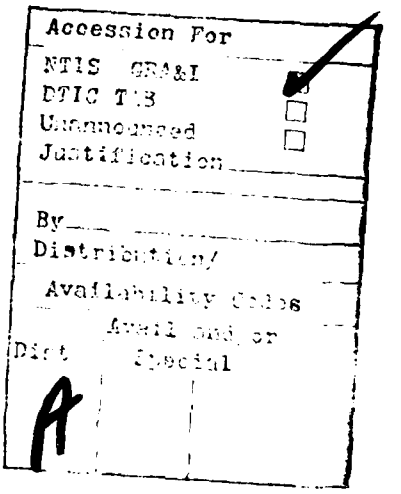
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PREFACE

This Final Report was prepared by General Electric Ordnance Systems, 100 Plastics Avenue, Pittsfield, Massachusetts, for Rome Air Development Center, Griffiss Air Force Base, New York, under contract F30602-80-C-0041.

It covers the period from January 1980 to December 1980. Mr. Regis C. Hilow, RBRA, was the RADC Project Engineer.

The work on this project was performed by the Electronic Circuits Engineering Operation and Components Engineering Unit. Project responsibility was held by Messrs. Thomas M. Ostrowski and Barney W. Hajduk of Circuit Design Engineering. Key individuals who made significant contributions to this report were Messrs. Bruce Newton, William Keller and Richard English.



CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION I

SUMMARY

This report details the efforts performed on a 16-bit microprocessor characterization contract for RADC. A general benchmark was developed and used to compare the performance of the 8086 and Z8000. A test pattern was developed and data was taken and analyzed to determine the operating region of the Z8001. AC and DC tests supplied by the vendor were analyzed and a preliminary analysis of the vendor's functional test was performed. A MIL-M-38510 slash sheet (not included in this report) was developed for the Z8001 and Z8002.

SECTION II

INTRODUCTION

This characterization was an extension of similar efforts performed for other microprocessors on previous RADC contracts.

A general benchmark was developed and used to compare the performance of the 8086 and Z8000. Section III of this report describes the results of this benchmarking effort.

A test pattern and program were developed for the Z8001 and a characterization was performed on a small sample of devices. Section IV describes the characterization and analysis of data.

The evaluation of the functional tests for the Z8001 and Z8002 was started on this contract. A list of tests required to check these devices was prepared and submitted to the vendor to assist in the evaluation. The vendor's AC and DC tests were evaluated and a MIL-M-38510 slash sheet was developed. Evaluation of the functional test will be completed on a future RADC contract. Section V of this report describes the review of the tests for the Z8001 and Z8002.

SECTION III

BENCHMARKING THE 8086 AND Z8001

OBJECTIVE

The objective of this evaluation was to develop a general benchmark for comparing the 8086 and Z8001 16-bit microprocessors. Since military applications include a large variety of microprocessor tasks, the benchmark had to provide a general assessment of each microprocessor's capabilities and a means of comparing them. It also had to be independent of programming/programmer bias and experience level.

SUMMARY

The microprocessors were benchmarked using five tasks and two mixes of instructions. In addition, the vendor support of the devices was examined.

The five tasks chosen were moving a block of data, adding and multiplying data arrays, sorting a data array, and servicing interrupts. The two mixes chosen were an arithmetic mix and a CRT terminal controller mix.

The results of the benchmarks are summarized in Table 3-1, with the performance of the Z8001 normalized to that of the 8086. The task benchmarks used the data calculated from operations on a 256 word array. A 5% deviation from calculated execution times was used for the 8086, since this is the minimum deviation to be expected, as detailed in the vendor's literature.

The benchmark results revealed that the Z8001 was significantly more efficient and significantly faster than the 8086 in six of the seven tests. The remaining test was inconclusive although the instruction prefetch mechanism of the 8086 could change this. The vendor of the 8086 states that actual execution times can be expected to be 5% to 10% slower than calculated execution times and this would increase the speed advantage of the Z8001.

Both devices were found to be well supported by vendor supplied development systems supporting high level language programming.

The Z8001 was found to be a better choice than the 8086 by these comparisons.

MICROPROCESSOR DESCRIPTION

The 8086 is an HMOS (high performance n channel) device which operates, in the military version, at a clock rate of 5 MHz. This device uses a dedicated register architecture and can address 1 megabyte of data memory.

TABLE 3-1 Z8001/8086 BENCHMARK PERFORMANCE SUMMARY

Task	Z8001 Execution Time	Z8001 Machine Cycles	Z8001 Bytes of Code
Move a block of data	64%	53%	91%
Add data arrays	83%	69%	89%
Multiply data arrays	61%	51%	71%
Sort a data array			
best case order	13%	81%	81%
worst case order	20%	17%	81%
Service interrupt	88%	97%	
Arithmetic mix	94%	77%	
CRT terminal controller	108%	86%	

1) Baseline is 8086 calculated performance, plus 5% per the vendor's literature.

2) 256 element arrays are assumed.

The Z8001 is an NMOS device which can operate at a clock rate of 4MHz and address 8 megabytes of data memory. It can utilize a Memory Management Unit to limit memory access, modify logical addresses to virtual addresses, and perform other memory management tasks.

Both devices support arithmetic operations on Binary Coded Decimal data, while the 8086 also supports ASCII arithmetic manipulations. Both devices have control lines used for multiprocessor applications.

BENCHMARKING OVERVIEW

The primary advantage of task benchmarking is that it illustrates the specific instructions available to the processor under test. However,the results obtained using this method can be misleading, since the selection of the task and programmer bias (in terms of preference, experience, and skill) can affect the results.

Mix benchmarking removes the human factor from the programming by the extraction of instruction types from a specific application. However, it can be limiting in that it uses only those instructions common to most microprocessors. The mix benchmark, therefore, does not compare the total instruction sets available to the microprocessors.

The general benchmark used both methods. Task benchmarks were used to compare the specific instructions available to each microprocessor, while the mix benchmarks compared the processors' common instructions. This combination of benchmarking methods yields an accurate measure of microprocessor performance.

All benchmarks assumed an ideal system with no delays caused by the operator, slow memory, or other causes.

Task Benchmarks

The tasks selected for the general benchmark are similar to many published benchmarks. These tasks were expanded to include referencing of the large memory addressable by the microprocessors, but are otherwise standard. The tasks included moving a block of data, addition and multiplication of data arrays, sorting of a data array, and servicing interrupts.

Coding of the tasks and derivation of the results can be found in Appendices A and B. Since the time required for execution of some of the tasks is dependent on the number of elements processed, the results are representative operating times calculated for these tasks using 256 element arrays. It was assumed that eight registers had to be saved and restored for servicing interrupts.

Mix Benchmarks

Two mixes representing contrasting tasks were chosen for the benchmark. The first mix was an arithmetic mix. This mix was derived from a military fire control simulation mix which was modified slightly to enable the testing of microprocessors rather than minicomputers. The mix modifications are described in detail in Appendix C. Floating point operations were deleted because they do not exist as a hardware feature on either processor. The detailed derivation of the results of this mix is presented in Appendices D and E.

A CRT terminal controller mix was developed to complement the arithmetic mix. A detailed description of the mix derivation is presented in Appendix F. An overview of the model system and overall task is presented here for convenience. Refer to Figure 3-1 for the system block diagram.

For the terminal mix, it was assumed that the microprocessor is totally dedicated to the modification and management of the system and video buffer memories. The system is interrupt driven with keyboard and video interrupts allowed. While the system memory is accessible at all times, the video buffer can be accessed only during vertical sync periods as determined by the video timer and controller and the control circuits.

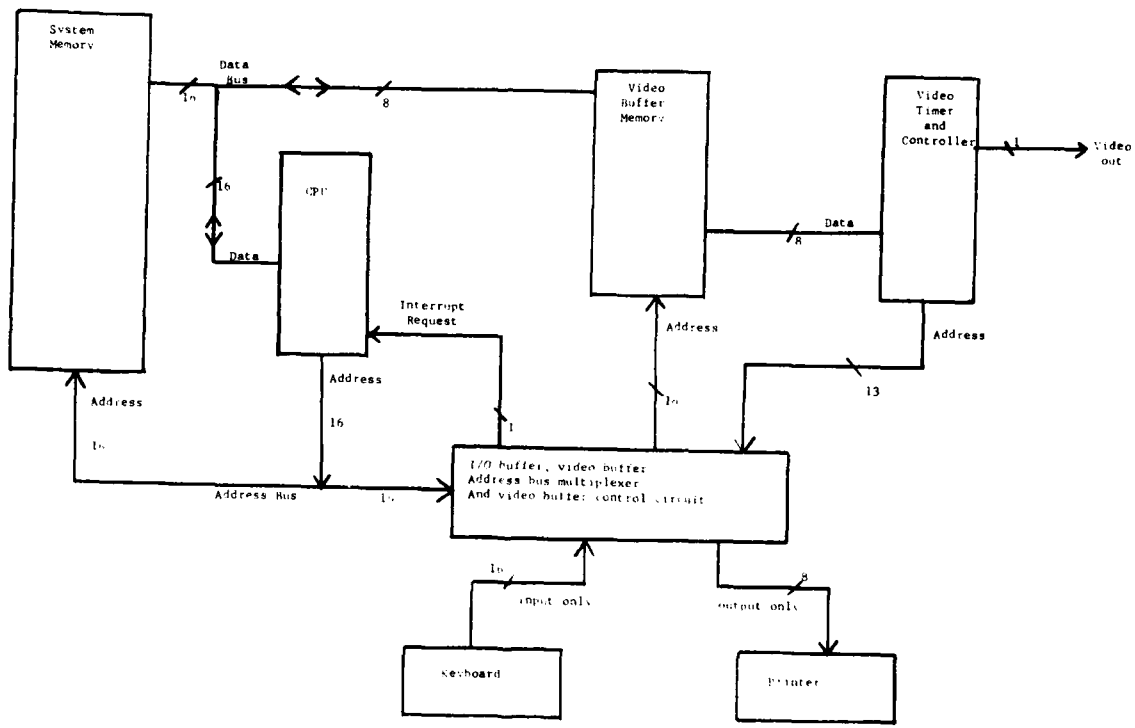


FIGURE 3-1 CRT TERMINAL CONTROLLER MODEL

The program developed was assumed to reside in system ROM thus limiting the addressing modes used. It was assumed that the processor was functioning in a "wait" loop prior to any interrupt and only those steps necessary for execution of the basic task were included in the definition. No "wait" states were required for memory access and all subroutines could be completed in one vertical sync period.

While the processor's time would not, in any reasonable application, be wasted on looping until interrupted as it is here, the system and programming were developed specifically to test data handling efficiency. This does not prohibit the execution of other tasks during the processor's "off" time, but comparing the processors is much clearer, and more meaningful, if added tasks are absent. The human interface was considered to be perfect for the task in the sense that it was not allowed to detract from the measurement of processor performance.

RESULTS

Benchmarking

The 8086 and Z8001 were compared using both task and instruction mix benchmarking methods.

Task benchmarking compared the lines of code and time required for the processors to execute specific tasks. This method allowed comparison of instructions not common to both microprocessors.

The five tasks chosen were moving a block of data, adding and multiplying data arrays, sorting a data array, and servicing interrupts. These tasks are similar to those used in many published microprocessor benchmarks, except for the modifications necessary to utilize the large memory space available to the processors.

The results of the task benchmark are summarized in Table 3-2 through 3-4, while the coding and derivation of these results can be found in Appendices A and B. For the purposes of comparison, representative execution times were calculated using the derived equations and assuming 256 element arrays.

Two mixes of instructions, derived from statistical analyses of instruction usage in specific microprocessor applications, were used to compare instructions common to both processors.

An arithmetic mix, derived from a military fire control simulation mix, was used to evaluate the arithmetic efficiency of the processors. The modifications to the military mix involved the deletion of floating point and transcendental functions, which are not hardware supported by either processor. The derivation of the mix can be found in Appendix C. A summary of the results for the arithmetic mix can be found in Table 3-5. The derivation of the results is included in Appendices D and E.

TABLE 3-2 TASK BENCHMARK RESULTS

	Z8001		8086	
Task	Bytes	Cycles	Bytes	Cycles
1. Move a block of 16 bit words	20	40 + 9n	22	35 + 17n
2. Add two arrays of 16 bit words, 16 bit result	40	47 + 54n	45	36 + 78n
3. Multiply two arrays of 16 bit words, 32 bit result	36	40 + 117n	51	36 + 230n
4. Sort an array of 16 bit words				
a) best case order	46	$9n^2 + 72n + 32$	57	$84n^2-13n+41$
b) worst case order	46	$9n^2 + 72n + 32+$ $(12(1+2+3...(n-1)))$	57	$91n^2-20n+41$
5. Service interrupt		306		421

TABLE 3-3 Z8001 TASK EXECUTION TIMES

Task *	Cycles	Time
1. Block move	2,344	.59 ms
2. Array addition	13,871	3.5 ms
3. Array multiplication	29,992	7.5 ms
4. Array sort		
a) best case order	608,288	152 ms
b) worst case order	999,968	250 ms
5. Service interrupt	306	.08 ms

* Calculations for tasks 1 through 4 are based on 256 element arrays.

TABLE 3-4 8086 TASK EXECUTION TIMES

Task *	Cycles	Calculated Time	Expected Run Time (+ 5%)
1. Block move	4387	.88 msec	.92 msec
2. Array addition	20,004	4.0 msec	4.2 msec
3. Array multiplication	58,916	11.8 msec	12.3 msec
4. Array sort			
a) best case order	5,501,737	1100 msec	1155 msec
b) worst case order	5,958,697	1192 msec	1251 msec
5. Service interrupt	421	.08 msec	.09 msec

* Calculations for tasks 1 through 4 are based on 256 element arrays.

TABLE 3-5 ARITHMETIC MIX (RESULT SUMMARY)

	Z8001 (4MHz)	8086 (5MHz)	
		Calculated	Expected (+5)
Group 1 Data Movement	16.2 msec	17.4 msec	18.3 msec
Group 2 Arithmetic	2.5 msec	3.3 msec	3.4 msec
Group 3 Shift/Rotate	.80 msec	.58 msec	.61 msec
Group 4 Compare	.22 msec	0.28 msec	.29 msec
Group 5 Branch Instructions	5.1 msec	5.4 msec	5.6 msec
Group 6 Index Register Operation	2.6 msec	1.2 msec	1.3 msec
Group 7 Logical Operations	.55 msec	.66 msec	.69 msec
Group 8 Input/Output	.22 msec	.14 msec	.15 msec
Total Time Required	28.2 msec	29.0 msec	30.0 msec

A CRT terminal controller mix compared the data handling capabilities of the processors. A detailed description of the derivation of the mix is included in Appendix F. The results of this benchmark are summarized in Tables 3-6 and 3-7. The symbols used in these tables are defined as follows:

N/A: Not Applicable

IM: Immediate

IR: Indirect Register

DA: Direct Address

Architectural and Software Considerations

In order to allow the easy translation of the earlier 8080 family software, the 8086 features a dedicated register architecture. While this system of register use is familiar to many, the regular register architecture of the Z8001 is much easier to learn and much faster to use in assembly language programming.

The Z8001 can directly support 8 megabytes of memory, 7 megabytes more than the 8086, and allows for direct conditional branching anywhere within the memory space. This address space is easily expandable to 48 megabytes by decoding the Z8001's status lines.

The 8086, while supplying unlimited range by the use of an unconditional jump command, is limited to +128, -127 bytes directly in conditional branching and loop instructions.

Due to the prefetched instruction queue, the penalty for taking a conditional branch is very high in the 8086. Instruction execution times are at least three, and sometimes four, times the basic instruction times. The use of the statistical determination that most conditional branches are less than +/- 127 bytes away aids in reducing the number of 8086 code lines. The Z8001, even though using two lines of code per branch instruction, is much faster in execution.

The ability to easily transform 8080 family code to 8086 code costs a great deal in performance. The decision to break with past processor architectures allows the Z8001 to be more efficient and more versatile than the 8086. While the loss of 8080 family software compatibility would be a small price to pay for the increase in performance, this loss is more apparent than real since the Z8001 is well supported by its software development system.

Product Maturity and Future

The 8086 is available in an evaluation board and a single board computer, as well as a separate IC.

The processor is supported by a development system which can support in circuit emulation. The development system will support Basic and Fortran, the proprietary PL/M, and the macro assemblers for the 8080 family as well as the 8086.

The 8089 Input/Output Processor and 8087 Arithmetic Coprocessor, which improve the performance of the 8086, are currently available.

While a second source agreement has been made, no devices are expected from the second source in the near future.

The Z8001 is also available as a single IC or in a single board computer or an evaluation board.

The development system supporting both the Z8001 and Z8002 offers optional in circuit emulation. The development system will support Fortran, Basic, Cobol, Pascal, the proprietary PL/Z, and the macro assemblers for the 8080 as well as Z80 families. Further software support is offered by a translation routine which will translate Z80, 8080, or 8085 source code into Z8000 source code.

Coprocessor development in support of the Z8000 family has been announced, but no details of function or availability have been released. The processors now being produced will, however, allow for the use of these coprocessors when they become available.

An active second source is currently producing the Z8000 family as well as marketing a development system.

TABLE 3-6 8086 CRT TERMINAL CONTROLLER MIX BENCHMARK RESULTS

Command	Address Mode	Machine Cycles	Number Used	Total Cycles
1. Clear Register (Word) 1/ (AND)	N/A	4	742	2,968
2. Clear Register (Byte) 1/ (AND)	N/A	4	750	3,000
3. Set/Load Byte	IM	4	679	2,716
4. Set/Load Word	IM	4	144	576
5. Move Byte, Register to Register	N/A	2	167	334
6. Move Word, Register to Register	N/A	2	1,157	2,314
7. Move Byte, Memory to Register	IR	14	107,520	1,505,280
8. Move Byte, Register to Memory	IR	15	119,040	1,785,600
9. Move Byte, Register to Memory (I/O) 3/	DA	12	11,999	143,988
10. Read Word, Memory to Register (I/O)	DA	10	12,384	148,608
11. Clear Memory (I/O) 2/	DA	11	12,384	136,224
12. Move # to Memory, Byte, (I/O) 3/	DA	12	148	1,776
13. Move # to Memory, Byte	DA	16	34,753	556,048
14. Move # to Memory, Byte	IR	15	1,233	18,495
15. Increment Register	N/A	2	203,744	203,742
16. Increment Byte Register	N/A	2	24,445	48,890
17. Decrement Register	N/A	2	1,575	3,150
18. ADD Register	IM	4	20	80
19. Logical AND to Register	IM	4	25	100
20. Subtract from Register	IM	4	4	16
21. Logical or to Register	IM	4	7	21
22. Clear/Set Bit I/O 4/	DA	12	45,537	546,444
23. Conditional Jump Taken	N/A	16	25,382	406,112
24. Conditional Jump Not Taken	N/A	4	153,280	613,120
25. Unconditional Jump	N/A	15	98,344	1,475,160

Total Cycles = 7,604,762

Total Calculated Time = 1.52 seconds

Expected "Execution" Time (+5%) = 1.60 seconds

NOTES:

(1) No CLEAR instruction exists for the 8086 microprocessor. CLEAR is implemented by an immediate AND with 0, word or byte as required.

(2) Clearing an I/O port is implemented by ANDing the accumulator with 0 and outputting the result to the required port. Thus 8 + 3 = 11 required cycles (AND AL, #0; OUT DX, AX).

(3) Movement of data to an output port requires an accumulator load followed by outputting the requested data, thus 8 + 4 = 12 cycles minimum (MOV B AX, #B; OUT DX, AX).

(4) Setting and clearing of output port bits is accomplished by setting or clearing the appropriate accumulator bits by an immediate MOV instruction, then outputting the results. Thus, 8 + 4 = 12 cycles are required (MOV AX, #set; OUT DX, AX).

TABLE 3-7 Z8001 CRT TERMINAL CONTROLLER MIX BENCHMARK RESULTS

	Command	Address Mode	Machine Cycles	Number Used	Total Cycles
1.	Clear Register, Word (CLR)	N/A	7	742	5,194
2.	Clear Register, Byte (CLRB)	N/A	7	750	5,250
3.	Set/Load Register, Byte (LDB)	IM	7	679	4,753
4.	Set/Load Register, Word (LD)	IM	7	144	1,008
5.	Move Byte, Register to Register (LDB)	N/A	3	167	501
6.	Move Word, Register to Register (LD)	N/A	3	1,157	3,471
7.	Move Byte, Memory to Register (LDB)	IR	7	107,520	752,640
8.	Move Byte, Register to Memory (LDB)	IR	8	119,040	952,320
9.	Move Byte, Register to Memory, I/O (OUT B)	DA	12	11,999	143,988
10.	Read Memory to Register, Word I/O	DA	12	12,384	148,608
11.	Clear Memory, Word 0 Out to I/O Port (Out)	DA	12	12,384	148,608
12.	Move # to Memory, Byte I/O (OUT B)	DA	12	148	1,776
13.	Move #, to Memory, Byte (LDB)	DA	14	34,753	486,542
14.	Move #, to Memory, Byte	IR	11	1,233	13,563
15.	Increment Register, Word	N/A	4	203,744	814,976
16.	Increment Register, Byte	N/A	4	24,445	97,780
17.	Decrement Register, Word	N/A	4	1,575	6,300
18.	Add Register, Byte (ADDB)	IM	7	20	140
19.	And Register, Byte (ANDB)	IM	7	25	175
20.	Subtract Byte (SUBB)	IM	7	4	21
21.	Logical OR Register, Byte (ORB)	IM	4	7	21
22.	Set/Clear Bit (SET B)	DA	16	47,537	760,592
23.	Conditional Jump, Taken (JP)	DA	8	25,382	203,056
24.	Conditional Jump, Not Taken (JP)	DA	8	153,280	1,266,240
25.	Unconditional Jump (JP)	DA	8	98,344	786,752

Total Cycles Required = 6,564,275

Total Time Required (4MHz) = 1.64 msec

SECTION IV

CHARACTERIZATION OF THE Z8001 MICROPROCESSOR

OBJECTIVE

The purpose of this effort was to perform a characterization of the Z8001 microprocessor. The following tasks were included in this effort:

1. Develop a short test pattern which would exercise a large percentage of the circuitry in the Z8001.

2. Develop a test program compatible with the Tektronix 3270 at RADC. This program would use a GO/NOGO functional test with worst case timing parameters.

3. Test commercial and military parts (if available) to determine device operating regions. Test results would provide an indication of the compliance of the device to vendor specified limits and whether the device will operate in the military temperature and voltage ranges.

SUMMARY

The characterization of the Z8001 determined device sensitivity to various combinations of V_{CC}, clock frequency, clock duty cycle, and logic level inputs over the -55 to +125°C temperature range. Test patterns were generated using a simple EPROM based Z8001 system developed for that purpose. A machine level program was written and transferred into the EPROM, a logic analyzer was connected, and the test vectors were recorded as the system ran. A test adapter and 3270 GO/NOGO functional test program were developed for taking and storing data. All of the programs that were developed are included in Appendix G.

Data was taken on seven commercial and two military parts. Devices were obtained from both manufacturers of the Z8001.

None of the devices passed at the vendor specified limits over the commercial voltage and temperature ranges. Vendor L's devices required that V_{IH} = 2.2 V and Vendor A's devices required that V_{IH} =2.3 V for 100% of the devices to pass. Vendor L's devices would operate only up to 3.5 MHz at 70°C with the minimum specified clock low time. Vendor L is aware of this duty cycle/temperature problem and is taking corrective action. Vendor A's devices did operate up to 4 MHz at 70°C but exhibited this problem at 125°C.

None of Vendor L's devices would operate over the entire military temperature and voltages ranges. At 125°C they all failed at V_{CC} = 4.5 V. Performance improved at 4.75 V but not all devices passed. The problem does not appear to be input level sensitive. Vendor L had not experienced this problem and offered to retest the devices. This may be done at a future date.

Two of Vendor A's devices passed over the military temperature and voltage ranges for some combinations of drive levels and frequencies. V_{IH} had to be 2.4 V or greater for any of the devices that did operate in this range. The maximum frequency they would work at over the entire range was 3.5 MHz. As previously mentioned they also exhibited the duty cycle/temperature problem at 125°C. Only one device passed at 125°C with V_{CC} = 4.5 V even with V_{IL} = 0.0 V. At V_{CC} = 4.75 V all devices passed if V_{IL} was less than 0.6 V indicating that the device was sensitive to input low drive levels.

As a result of this effort, test patterns and programs are in place on a Tektronics 3270 and are available to perform additional testing or more extensive characterization.

DISCUSSION

The Z-8001, introduced in 1979, is a radical departure from the dedicated register architecture of the earlier eight bit machines. The use of a sophisticated architecture featuring sixteen bit, non-dedicated registers, plus a set of parallel stack registers, greatly increases the difficulty in developing an effective test.

Vector Development

Part of the characterization effort consisted of the development of a short pattern that could be used to test the Z8001. Since a large portion of the circuitry in a microprocessor can be tested by the fetching and execution of a small number of instructions, the use of a short pattern provides a good indication of device performance while minimizing the number of pattern loads required during the test. The following methods of vector development were investigated:

1. Manual generation of the test vectors.

2. The hardware emulation approach to vector generation which includes the use of the 3270 to record the test vectors.

3. The manual extraction of the test vectors from an operating Z8001 system.

The first choice was ruled out because information was not available in the vendors' literature to indicate on what clock cycles instructions and data had to be available to the processor. The second was ruled out due to the risk and cost involved in using a Z8001 board which had just come on the market. The third method which uses a logic analyzer to extract the test vectors from an operating system was chosen.

Vector Generation

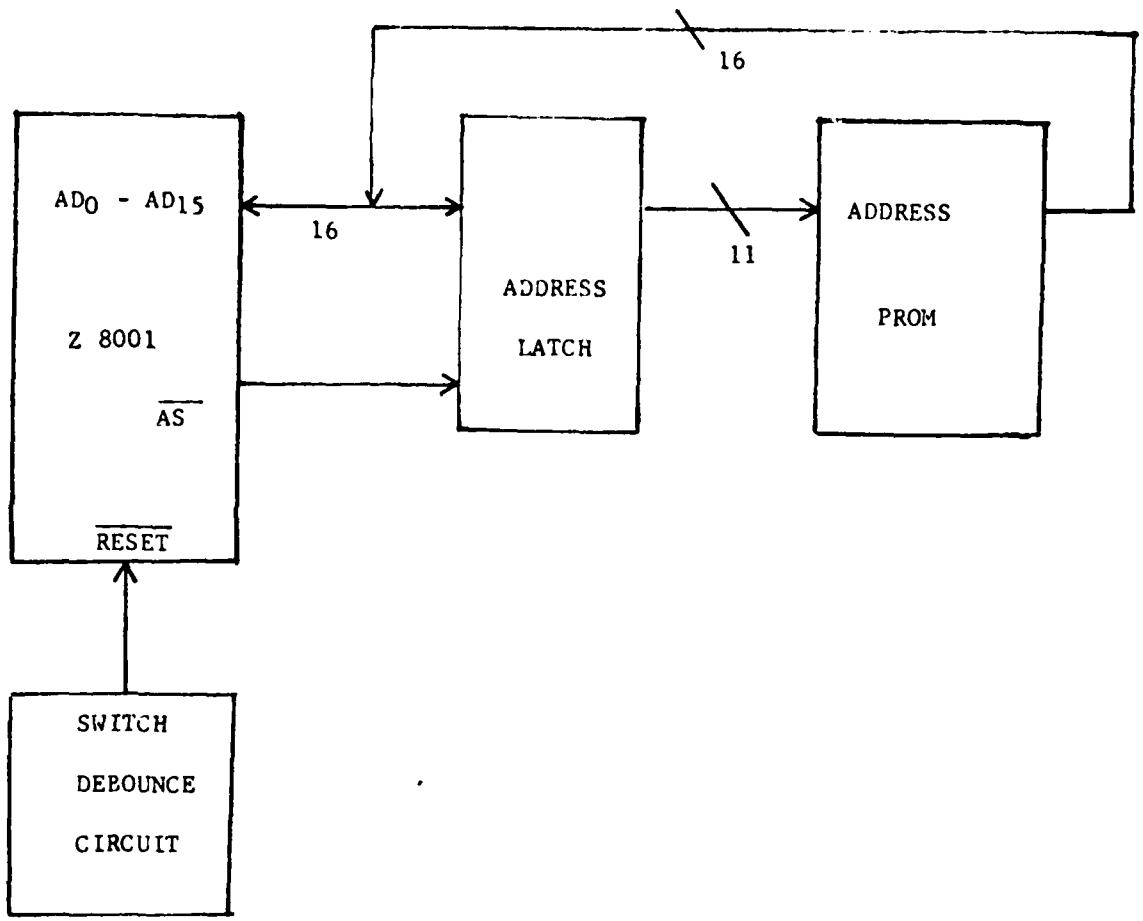
The system shown in Figure 4-1 was designed and built. A machine language program was written and programmed into the EPROMs. After the program was debugged, a logic analyzer was connected and the vectors were extracted and recorded for later transcription to the 3270.

Vector Description

The pattern developed contains 512 vectors. The following functions of the Z8001 were tested:

1. Reset
2. Moving data into and out of all user accessible registers, including the refresh, NPSAP, and normal stack pointers, at least once.
3. Multiple "automatic" register loads to and from memory.
4. Conditional and unconditional jumps.
5. The clearing of word and long word registers.
6. Add and multiply.
7. Test word and long word registers.
8. Output word operands
9. Halt

These functions used intersegment data and code fetches and five of the eight addressing modes.


16
AD0 - AD15
16
ADDRESS
LATCH
11
ADDRESS
PROM
Z 8001
AS
RESET
SWITCH
DEBOUNCE
CIRCUIT

3270 Test Description

The Z8001 test utilized a GO/NOGO functional test in which PASS/FAIL information was recorded as the temperature, logic input levels, power supply voltage, operating frequency, and clock duty cycle were varied over the following ranges:

1. Temperature
   -55, 0, 25, 70, 125°C

2. Logic Input Levels
   V_{IL} held at 0.0 V, V_{IH} varied from 2.0 V to 2.5 V in 0.1 V steps.
   V_{IH} held at 3.5 V, V_{IL} varied from 0.4 V to 0.8 V in 0.1 V steps.

3. Power Supply Voltage
   4.25 V to 5.75 V in 0.25 V steps

4. Operating Frequency
   250 KHz and 500 KHz to 6 MHz in 500 KHz steps

5. Clock Duty Cycle
   20% to 80% in 10% steps

Duty cycle was used as a variable parameter during the test to make testing more convenient. The clock parameters are not specified as a function of duty cycle as they are for some processors. A minimum clock high and low time are specified for the Z8001. At 4 MHz, these equate to a 50% duty cycle. By varying the duty cycle it is possible to vary the clock high and low times.

During the test, the clock input high and low voltages were maintained at the vendor specified limits of V_{CC}-0.4 V and 0.45 V, respectively. The output comparison levels were 2.4 V and 0.4 V which are the vendor specified limits for V_{OH} and V_{OL}, respectively.

The test pattern was run in five passes to ensure that output timing was checked at the manufacturer specified delays. The first pass checked $\overline{AS}$ and $\overline{DS}$ read, the second checked $\overline{DS}$ write, the third checked $\overline{DS}$ I/O, the fourth checked $\overline{MREQ}$, and the fifth checked the address/data bus (AD0 to AD15).

Figure 4-2 shows a high level flow chart of the test program.

The load circuit shown in Figure 4-3 was connected to the device output pins. It provides 100% capacitive loading (including 3270 capacitance), 70% resistive loading for V_{OL} and 80% resistive loading for V_{OH}. With the load connected the high impedance voltage floats to a value between 1.0 V and 2.0 V. This allows checking of the high impedance state during functional test.

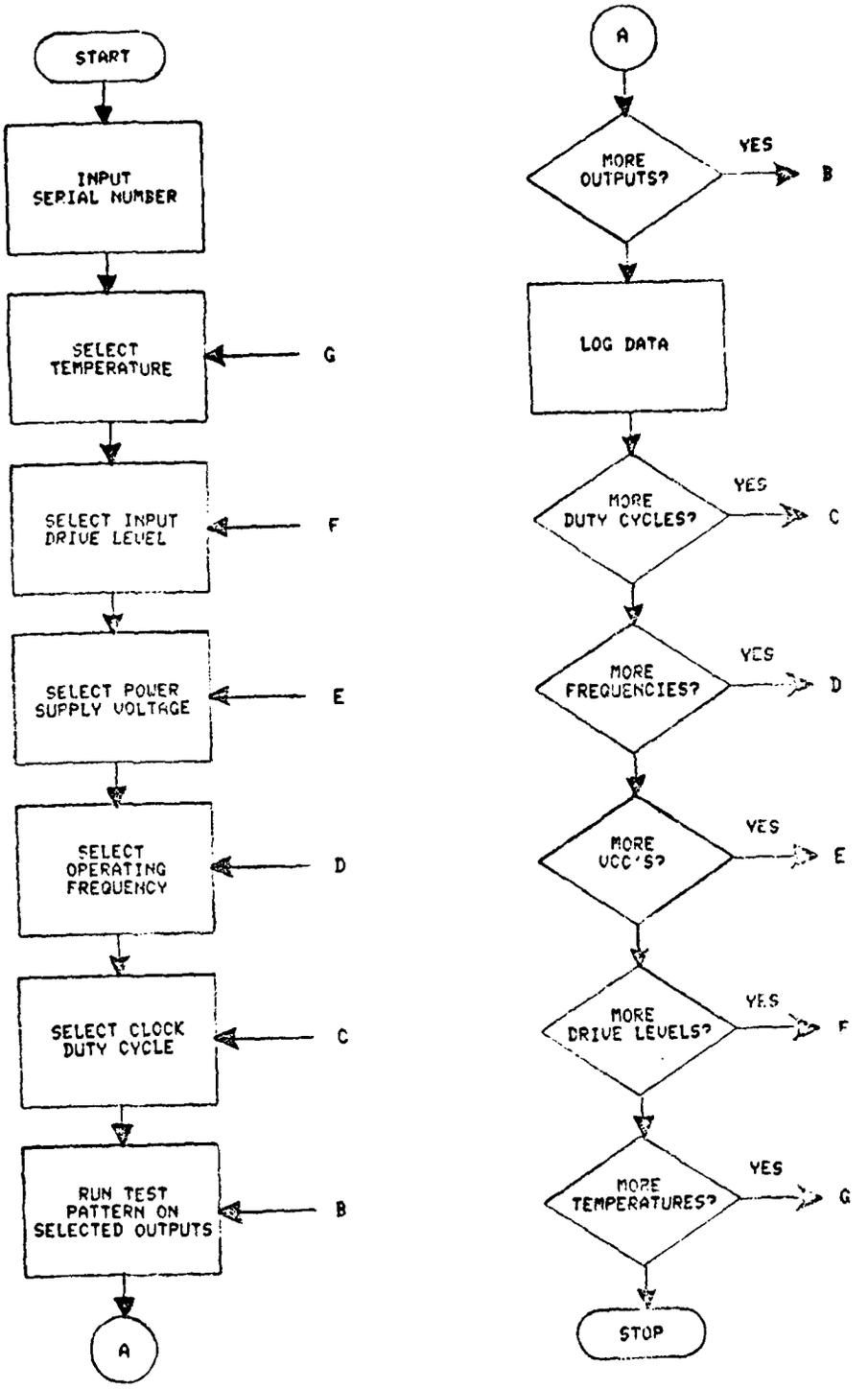


FIGURE 4-2 Z8001 TEST PROGRAM FLOW CHART

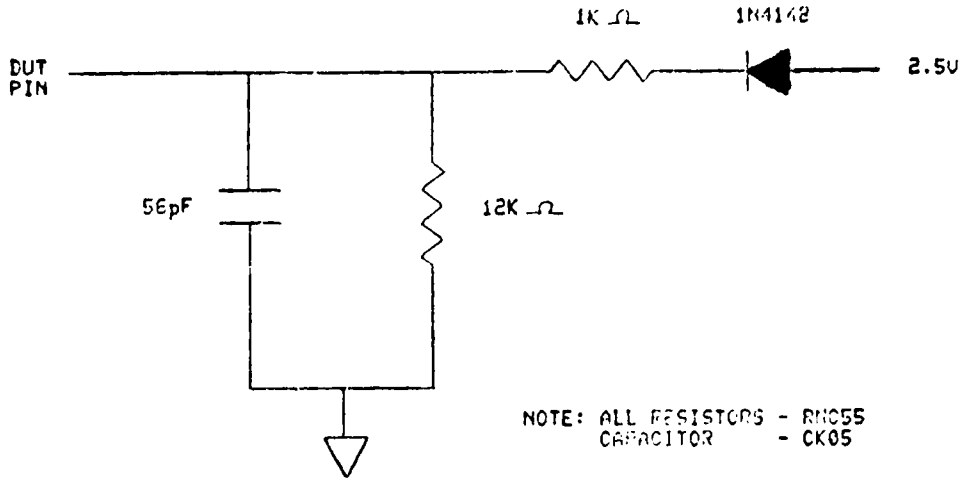


FIGURE 4-3 Z8001 LOAD CIRCUIT

DATA ANALYSIS

Five parts from Vendor L and four from Vendor A were tested. All parts received for characterization were serialized as indicated in Table 4-1.

TABLE 4-1 DEVICE SERIAL NUMBERS

Device Number	Vendor	Mark Step	Type	Date Code
1	L	V	Commercial	8043
2	L	V	Commercial	8043
3	L	V	Commercial	8104
4	L	V	Commercial	8104
5	L	V	Commercial	8104
6	A	W	Military	8051
7	A	W	Military	8051
8	A	W	Commercial	8052
9	A	W	Commercial	8052

Vendor L was able to provide V step devices. The V mask step represents the latest and first fully functional version of the Z8001 and is the version that will be qualified. The W mask step parts from Vendor A are one revision earlier. They have a few functions that do not operate properly. However, these were not included in the test pattern that was developed so that any failures that occurred could not be attributed to them.

Vendor L could not supply military grade devices because they were just updating their test program to permit testing over the military temperature range. The two military grade parts from Vendor A were received as samples since they were not yet marketing their military devices.

The same chip is used for the commercial and military grade devices. Since the only distinction between the two grades is the temperature and power supply ranges over which the parts will operate, all devices were tested over the full commercial and military ranges.

Since a large amount of data was taken and analyzed it is not possible to include all of the shmoo plots that were generated in this report. Figures 4-4 through 4-7 summarize device operation over the commercial range and Figures 4-8 through 4-9 summarize operation over the military range. A 50% duty cycle was used to guarantee the minimum clock high and low times at 4 MHz. Figures 4-4, 4-6, and 4-8 show the number of devices that passed as frequency and V_{IH} were varied. For these figures, V_{IL}=0.0 V. Figures 4-5, 4-7, and 4-9 show the number of devices that passed as frequency and V_{IL} were varied. For these figures, V_{IH} = 3.5 V. The solid lines on these plots indicate the specified operating regions.

From Figures 4-4 and 4-5 it can be seen that Vendor L's parts did not operate over the specified commercial range. Examining the results up to 3.5 MHz it can be seen that V_{IH} had to be raised to 2.1 V for 80% of the devices to pass and 2.2 V for 100% to pass. Operation with V_{IL} = 0.8 V was not a problem. The failures at 4 MHz were duty cycle/temperature, and not drive level, related. This will be explained in more detail later.

Figures 4-6 and 4-7 show the performance of Vendor A's parts over the commercial operating range. From these figures it can be seen that V_{IL} was not a problem and that the devices would operate at 4 MHz. Vendor A's devices did not exhibit the duty cycle/temperature problem at 70°C. It can also be seen that V_{IH} had to be raised to 2.3 V for 100% of the devices to pass.

Figures 4-8 and 4-9 show the performance of Vendor A's parts over the military operating range. From these figures it can be seen that two of the four devices tested passed for some combination of drive levels and frequency. It should be noted that of the two devices that passed, one was commercial and one was military. The other military part (7) did not operate over the entire voltage range at 125°C. It can be seen from the figures that V_{IH} had to be 2.4 V or greater for any of the devices to operate and even then the maximum frequency was limited to 3.5 MHz. Operation at higher frequencies was again related to the duty cycle/temperature problem.

No summary plots are included for Vendor L's devices for the military operating region since none of them operated over the entire voltage and temperature ranges. The problems that were encountered are explained below.

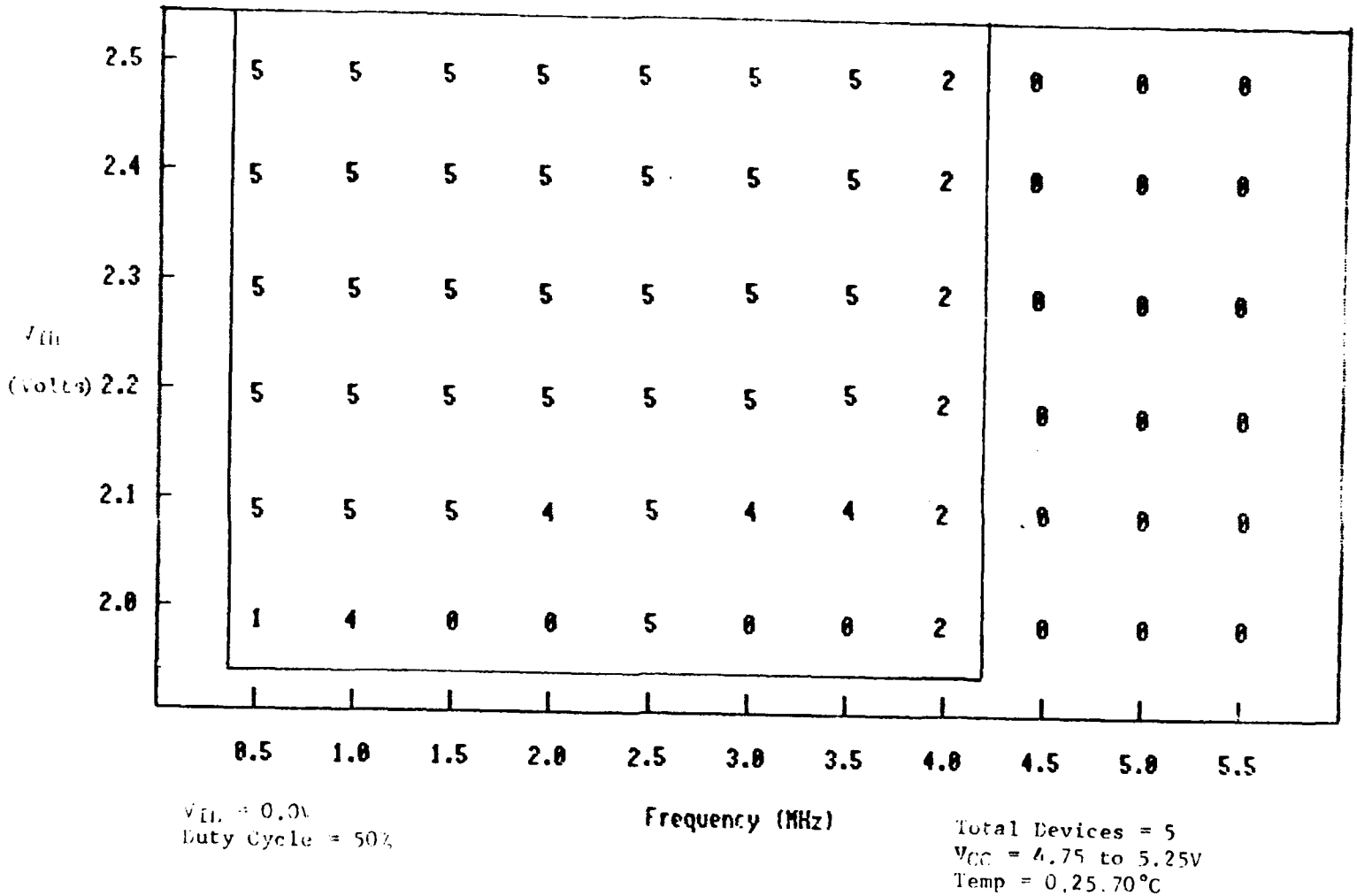


FIGURE 4-4 COUNT OF VENDOR L'S PASSING DEVICES FOR COMMERICAL LIMITS

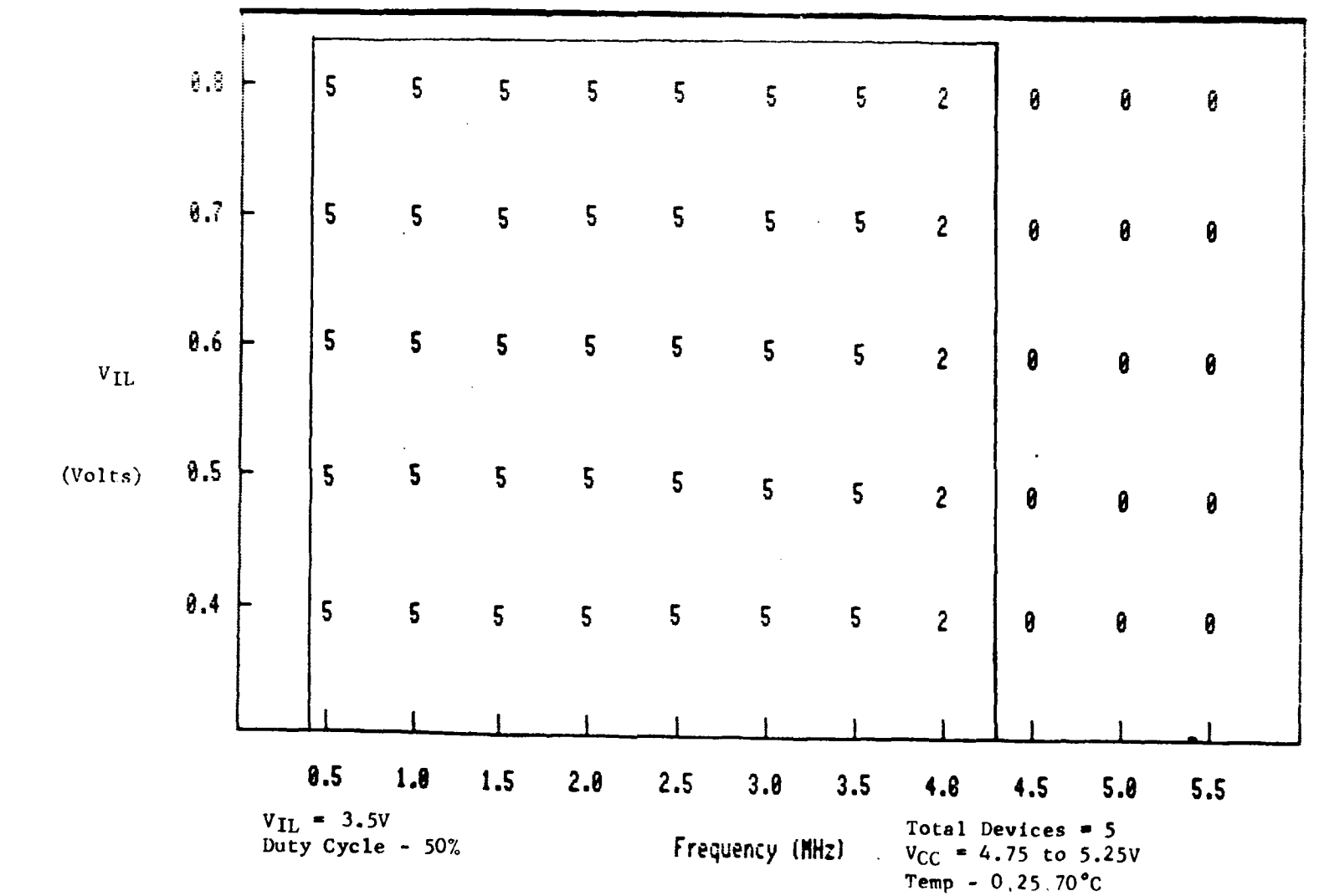
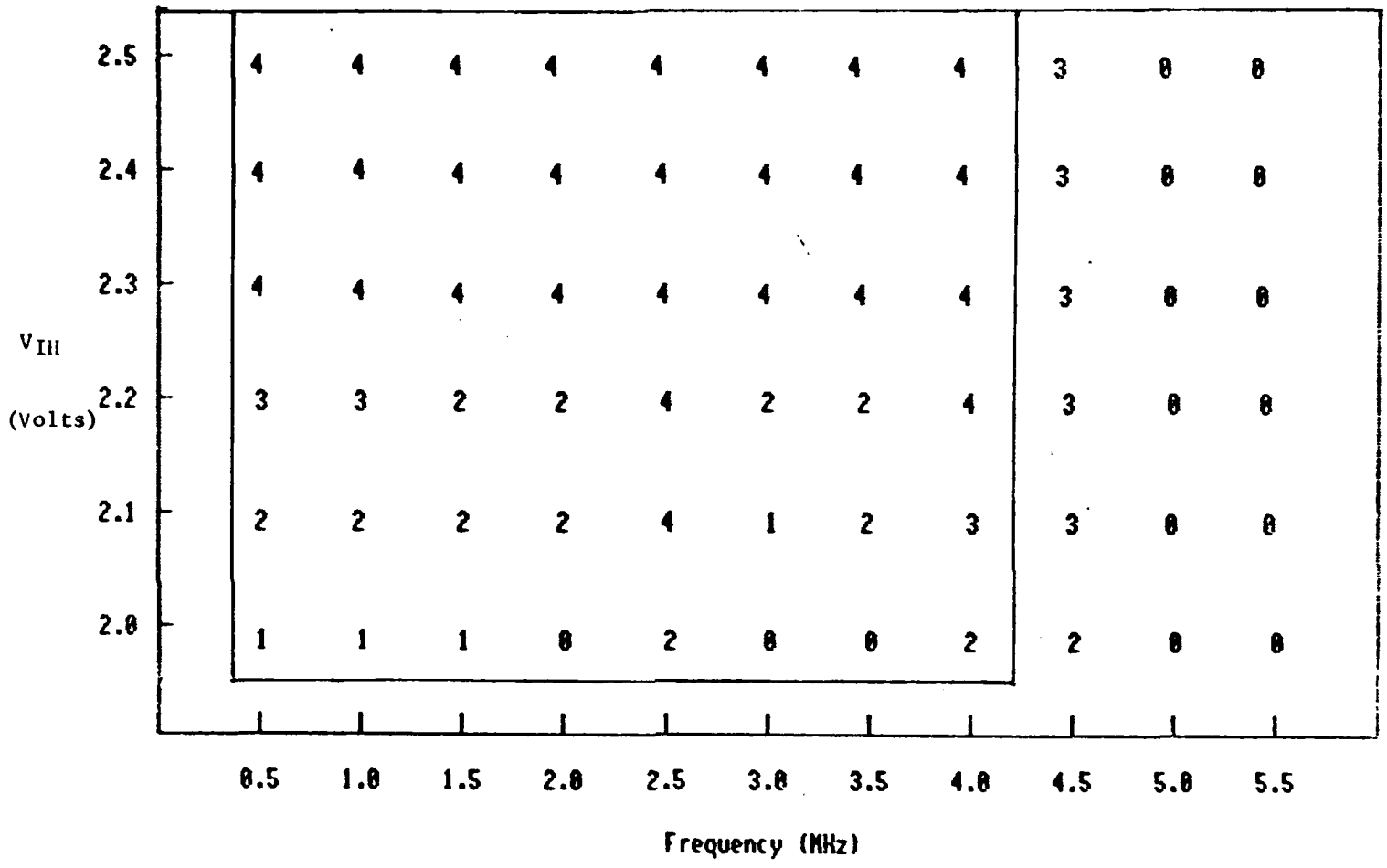


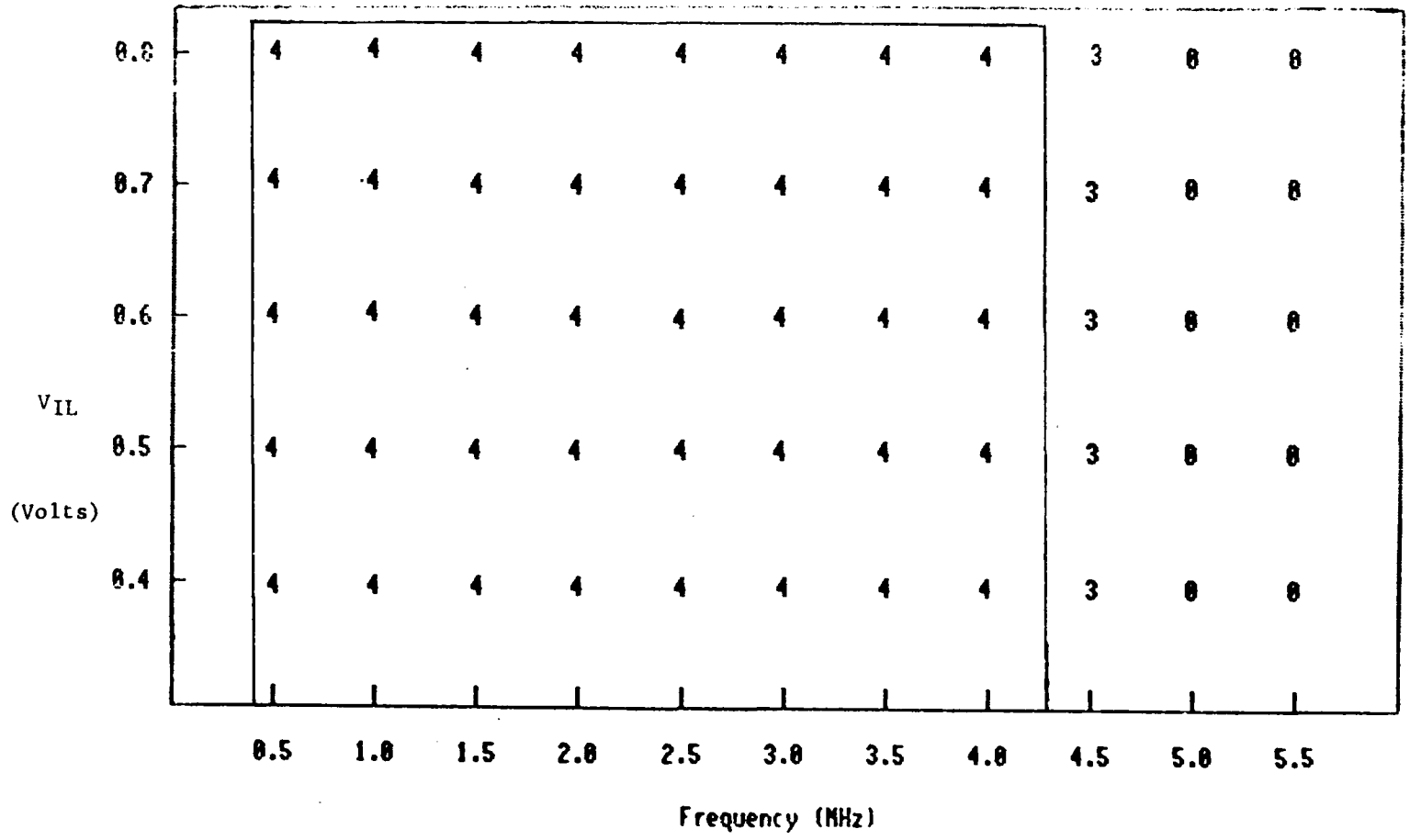
FIGURE 4-5 COUNT OF VENDOR L'S PASSING DEVICES FOR COMMERCIAL LIMITS



V_{IL} = 0.0V
Duty Cycle = 50%

Total Devices = 4
V_{CC} = 4.75 to 5.25V
Temp = 0,25,70°C

FIGURE 4-6 COUNT OF VENDOR A'S PASSING DEVICES FOR COMMERCIAL LIMITS



V_{IH} = 3.5V
Duty Cycle = 50%

Total Devices = 4
V_{CC} = 4.75 to 5.25V
Temp = 0 25,70°C

FIGURE 4-7 COUNT OF VENDOR A'S PASSING DEVICES FOR COMMERCIAL LIMITS

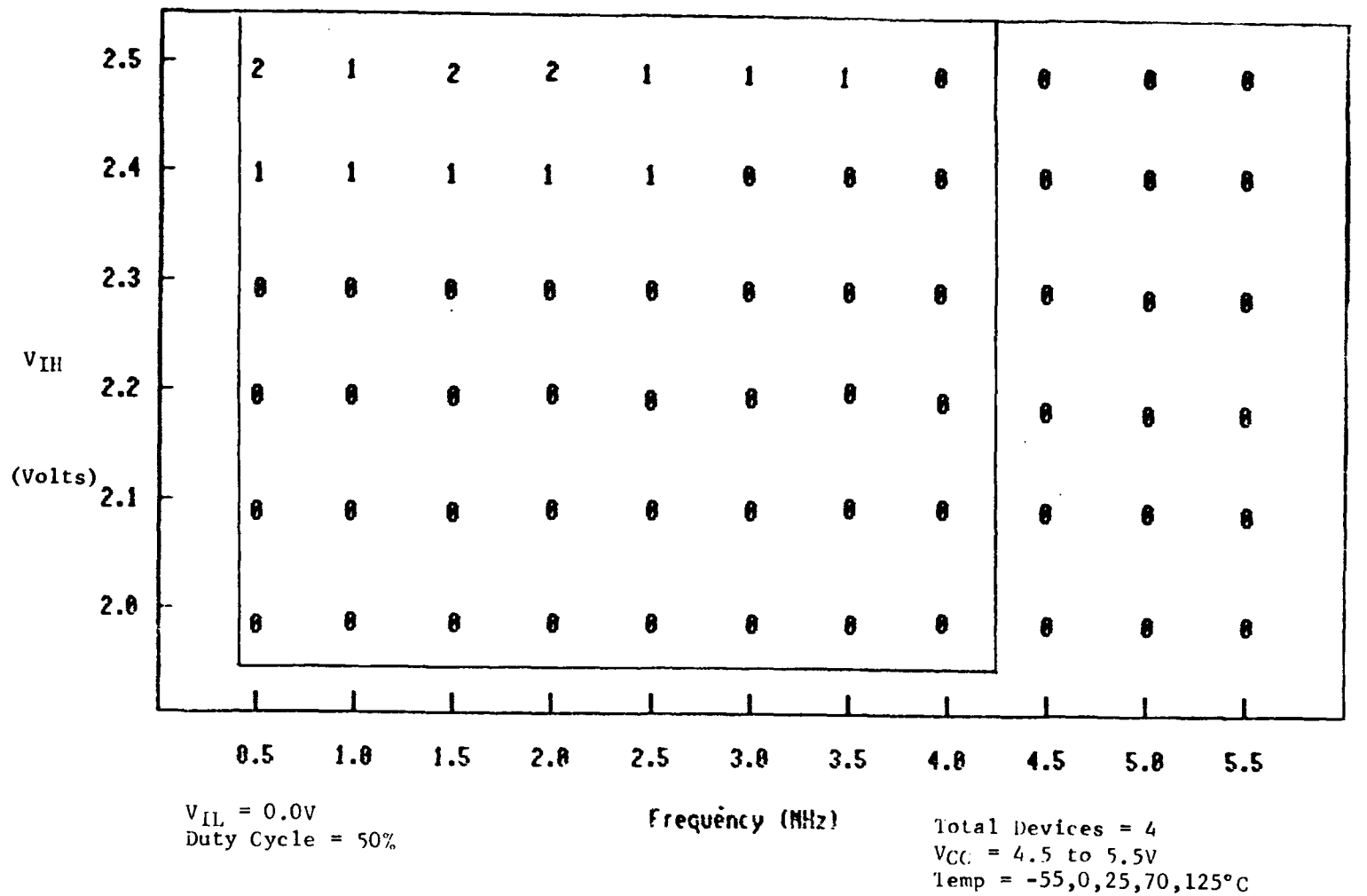


FIGURE 4-8 COUNT OF VENDOR A'S PASSING DEVICES FOR MILITARY LIMITS

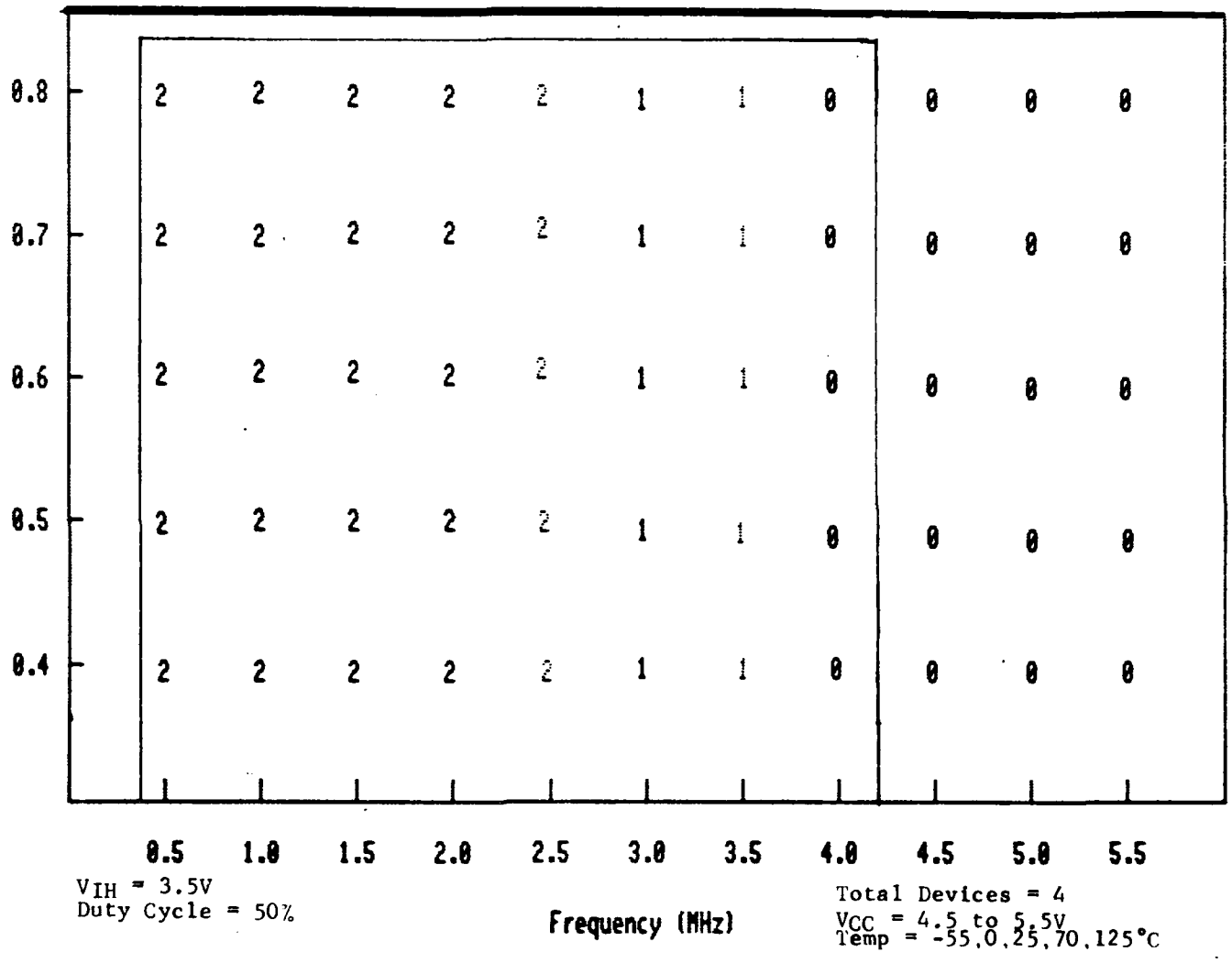


FIGURE 4-9 COUNT OF VENDOR A's PASSING DEVICES FOR MILITARY LIMITS

Additional shmoo plots are included to illustrate some of the results obtained. Most of the plots are for Vendor L's devices. Vendor A's devices operated in a similar manner. Where significant differences in performance occurred, plots are included for Vendor A's devices also.

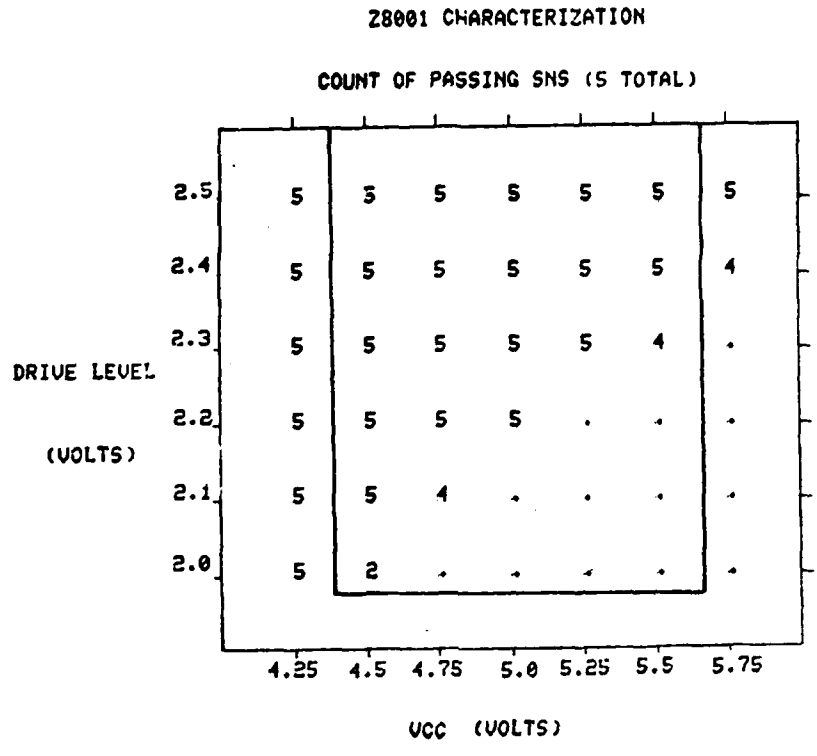
Figure 4-10 shows the relationship between V_{IH} and V_{CC} at -55°C, 4 MHz, and 50% duty cycle. It can be seen that as V_{CC} increases, V_{IH} has to increase for the device to pass. This is to be expected since the threshold for NMOS devices is highest at low temperatures and high values of V_{CC}. It should be noted that V_{IH} has to be at least 2.3 V for 80% of the devices to pass and 2.4 V for all devices to pass. Vendor A's devices performed in a similar manner and also required 2.4 V for all devices to pass.

Figure 4-11 shows the relationship, for Vendor L's devices, between V_{IL} and V_{CC} at 125°C, 4 MHz, and 50% duty cycle. Since low V_{CC} and high temperature are the worst case conditions for V_{IL} one might assume that the failures at 4.5 and 4.75 volts could be eliminated if V_{IL} were less than 0.4 V. However, Figure 4-12 shows that the devices failed even when V_{IL} was 0.0 V. Vendor L was contacted to determine whether they had ever experienced this problem. They said that they had not and they did not know what might cause it. They did offer to retest the devices to see if they obtained similar results. This option may be exercised at a future date. Figure 4-13 is the same as Figure 4-11 except that it is for Vendor A's devices. It can be seen that only one of Vendor A's devices passed for all V_{IL} values at 4.5 V and 125°C. This would indicate that the problem was not threshold related since the failing devices did not pass even when V_{IL}=0.0 V. However, at V_{CC}=4.75 V, all devices passed when V_{IL} was less than 0.6 V. This does indicates that a threshold problem might exist at high temperature and low V_{CC}. Additional investigation in this area is recommended.

Figure 4-14 is a plot of frequency versus temperature at 50% duty cycle, V_{CC} = 5.0 V, V_{IH} = 3.5 V, and V_{IL} = 0.8 V. This plot is for nominal V_{CC} so that the V_{IH} and V_{IL} values used would not be affected by the variation in temperature. In examining the figure, one can see that the maximum frequency at which the device operates decreases with increasing temperature. Normally this is expected since NMOS slows down with increasing temperature. However, in this case the device operates only up to 3 MHz even though it is specified to operate at 4 MHz.

Figure 4-15 is the same as Figure 4-14 except that the duty cycle was reduced to 40%. In this plot a definite improvement in performance is seen at higher temperatures. All devices now operate at 4 MHz, at 70 and 125°C. However, low temperature performance is now sacrificed.

Figures 4-16 and 4-17 are plots of duty cycle versus frequency at -55 and 125°C, respectively. V_{CC} and the input drive levels are the same as in Figures 4-14 and 4-15. In comparing Figures 4-16 and 4-17, it can be seen that device performance is significantly degraded for the higher duty cycles at 125°C. The area to the left of the solid lines indicates

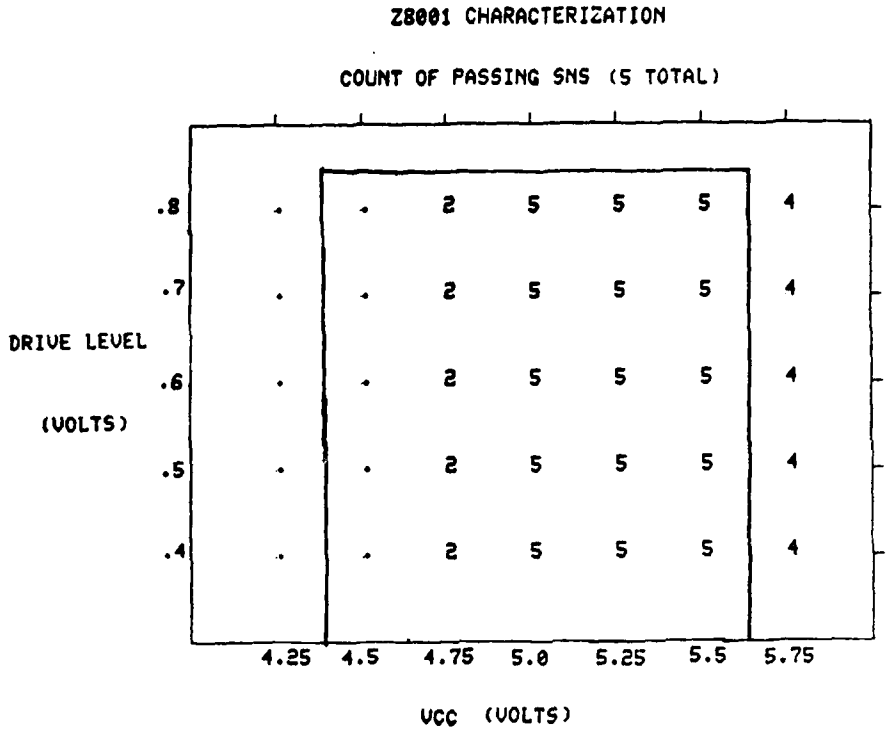


V_{IL} = 0.0V
Frequency = 4 MHz
Duty Cycle = 50%

1. V_{IH} has to increase with increasing V_{CC} for the devices to operate at -55°C. These are worst case conditions for V_{IH}.

2. Vendor A's devices performed in a similar manner.

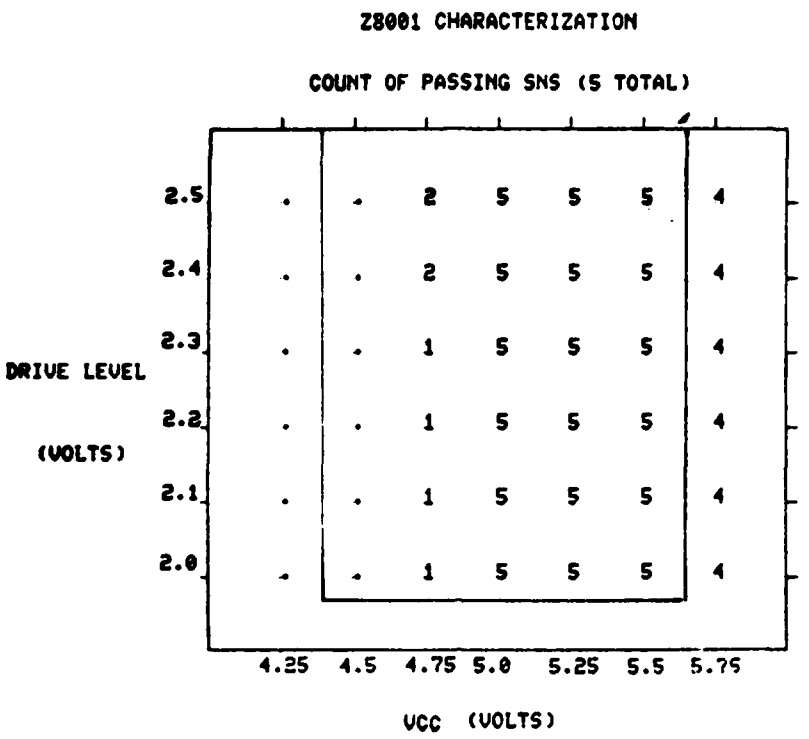
FIGURE 4-10 V_{IH} VS. V_{CC} AT -55°C FOR VENDOR L DEVICES



V_{IH} = 3.5 V
Frequency = 4 MHz
Duty Cycle = 50%

1. Device performance deteriorated if V_{CC} was less than 5V. Low V_{CC} and high temperature are worst case conditions for V_{IL}. Problem does not appear to be threshold related however because devices did not pass even when V_{IL} was 0.0V. (See Figure 4-12).

FIGURE 4-11 V_{IL} VS. V_{CC} AT 125°C FOR VENDOR L DEVICES

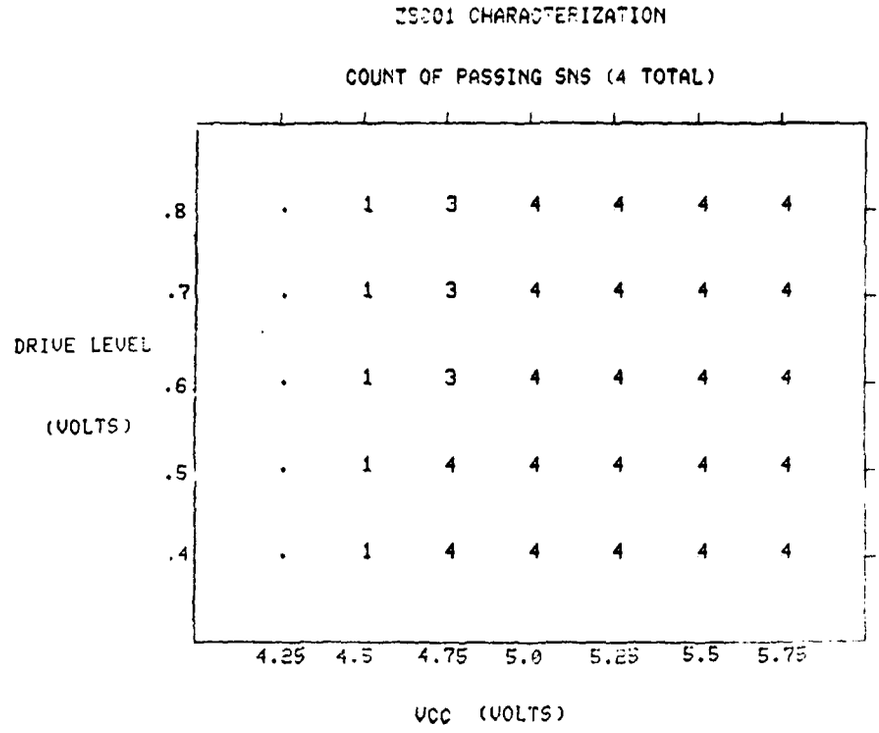


V_{IL} = 0.0V
Frequency = 4 MHz
Duty Cycle = 40%

1. None of Vendor L's devices passed at V_{CC} = 4.5V indicating that V_{IL} threshold was not the cause. At 4.75V two devices passed if V_{IH} was greater than 2.4V. Vendor L could not explain reason for this trend. Low V_{CC} and high temperature are best case conditions for V_{IH}.

2. One of Vendor A's devices passed at all V_{IH} drive levels for V_{CC} = 4.5V and all four passed at V_{CC} = 4.75V.

FIGURE 4-12 V_{IH} VS. V_{CC} AT 125°C FOR VENDOR L DEVICES

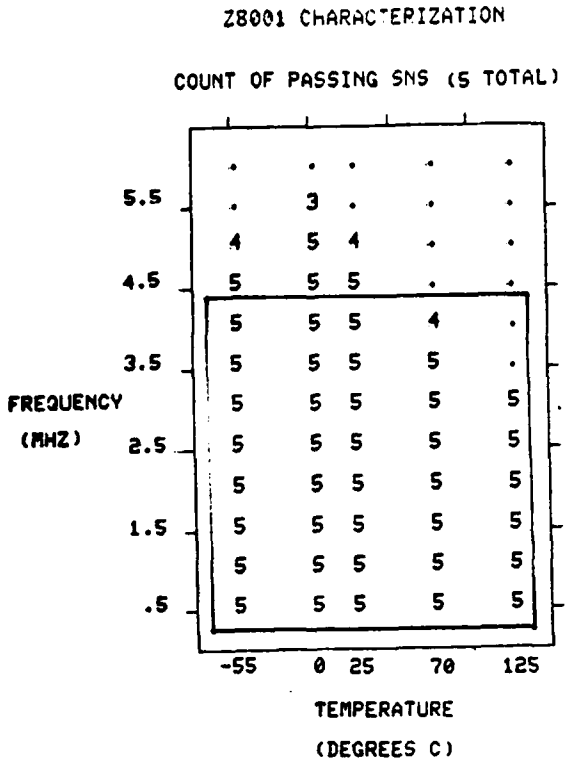


V_{IH} = 3.5 V
Frequency = 4 MHz
Duty Cycle = 50%

1. Only one device passed when V_{CC} = 4.5V. This was also true when V_{IL} was 0.0V and V_{IH} was varied.

2. At V_{CC} = 4.75, all devices passed when V_{IL} was less than 0.6V indicating a threshold problem might exist.

FIGURE 4-13 V_{IL} VS. V_{CC} AT 125°C FOR VENDOR A DEVICES.



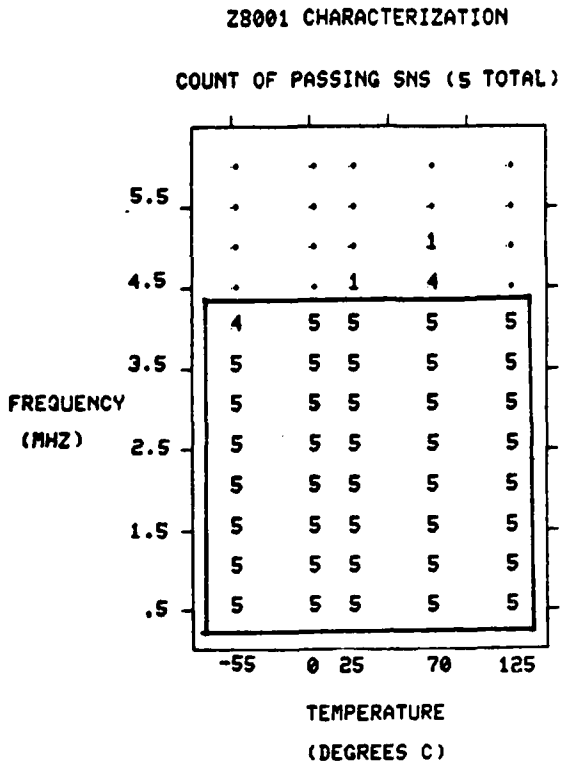
V_{IL} = 0.8V
V_{IH} = 3.5V
V_{CC} = 5.0V
Duty Cycle = 50%

1. Nominal V_{CC} was used to minimize threshold effects.

2. Maximum operating frequency decreases with increased temperature. Vendor L's devices operate up to 3 MHz.

3. Vendor A's devices also dropped out at 3 MHz at 125°C but passed up to 4 MHz at 70°C.

FIGURE 4-14 FREQUENCY VS. TEMPERATURE AT 50% DUTY CYCLE FOR VENDOR L DEVICES



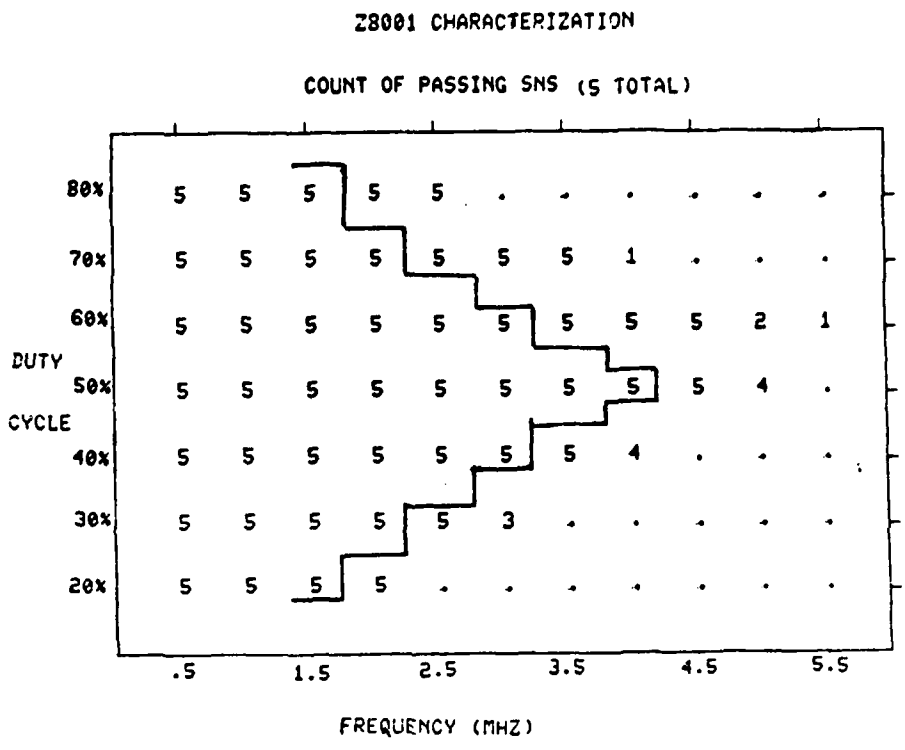
V_{IL} = 0.8V
V_{IH} = 3.5V
V_{CC} = 5.0
Duty Cycle = 40%

1. Nominal V_{CC} was used to minimize threshold effects.

2. At 40% duty cycle, all of Vendor L's devices passed to 4 MHz at high temperatures but one device dropped out at -55°C. A longer clock low time is required at high temperatures but it deteriorates low temperature performance.

3. Vendor A's devices also performed better at 40% duty cycle and 125°C with all four passing up to 4 MHz.

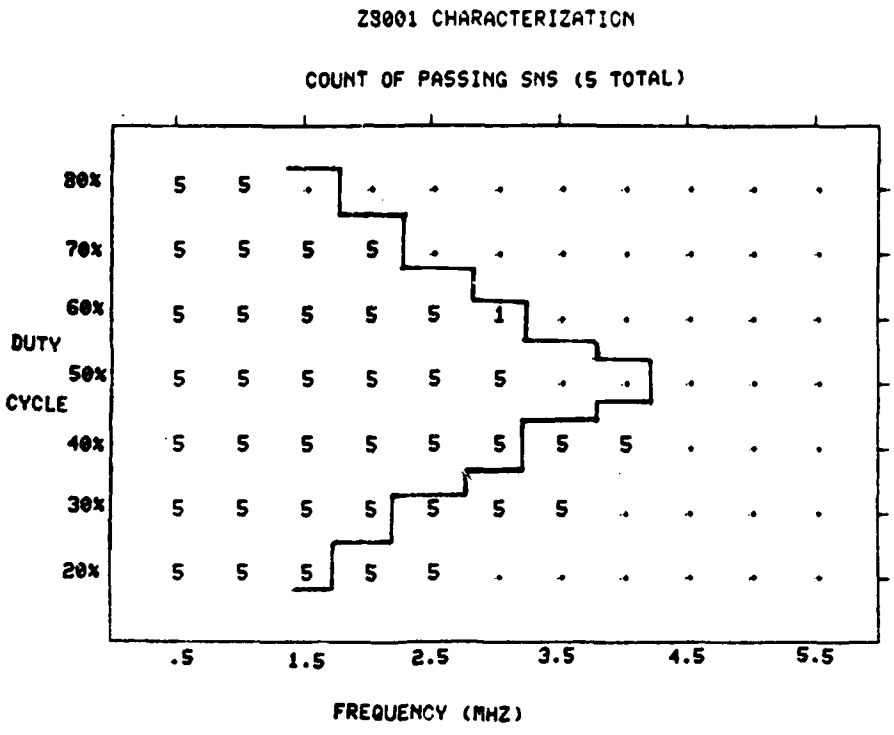
FIGURE 4-15 FREQUENCY VS. TEMPERATURE AT 40% DUTY CYCLE FOR VENDOR L DEVICES



V_{IL} = 0.8V
V_{IH} = 3.5V
V_{CC} = 5.0V
Temp = -55°C

1. Nominal V_{CC} was used to minimize threshold effects.
2. All of Vendor L's devices passed within the specified range.
3. All of Vendor A's devices passed within the specified region also.
4. At low temperature, both vendors' devices operated better with shorter clock low times (higher duty cycles).

FIGURE 4-16 DUTY CYCLE VS. FREQUENCY AT -55°C FOR VENDOR L DEVICES



V_{IL} = 0.8V
V_{IH} = 3.5V
V_{CC} = 5.0V

TEMP = 125°C

1. Nominal V_{CC} was used to minimize threshold effects.
2. At high temperature, Vendor L devices are sensitive to clock low time and failures occur.
3. Vendor A's devices exhibited the same problem.

FIGURE 4-17 DUTY CYCLE VS. FREQUENCY AT 125°C FOR VENDOR L DEVICES

the duty cycles for which the minimum clock high and low times are met at each frequency. In Figure 4-15 it can be seen that all devices pass within the specified range. However, at 125°C failures occurred at 1.5 MHz and above 3 MHz. As previously mentioned this problem limited 70°C operation for Vendor L's parts to 3.5 MHz. Vendor A's parts exhibited the same problem at 125°C but not at 70°C. Vendor L was questioned about the problem and indicated that they were aware of it. The Z8001 is a dynamic device and the clock low time is used for charging of dynamic nodes. At high temperatures parts fail when operated at the minimum clock low time. The problem is instruction dependent and existed in previous versions of the device. The product engineer did not have available the list of instructions which were affected. He did indicate that the problem was remedied somewhat with the last mask change. Vendor L is planning a die shrink and indicated that the problem should be eliminated with it.

A 6 MHz version of the Z8001 will also be qualified. Some of the devices did pass at 5.5 MHz but not over the entire commercial or military ranges. It should be noted that the parts tested were sold or sampled as 4 MHz devices.

A minimum frequency of .5 MHz is specified for the device. The parts were tested down to .25 MHz. It was found that if a device operated at .5 MHz for a given set of conditions then it would also operate at .25 MHz at these conditions.

CONCLUSIONS AND RECOMMENDATIONS

The following conclusions were reached as a result of the characterization:

1. Both vendors' parts exhibited similar performance characteristics for any given set of conditions.

2. The devices are sensitive to the V_{IH} level and this limit may have to be relaxed. Vendor A's devices required a slightly higher logic one level than Vendor L's over the commercial range. Both vendors' devices required a logic one level of at least 2.4 V over the military range.

3. Operation at 4 MHz is a problem at high temperature. This is caused by the limit on clock low time. Operation at 6 MHz will also be a problem since the clock low time is even less.

4. The .5 MHz minimum frequency limit is conservative.

5. Neither vendors' devices would operate well at 125°C when V_{CC} was less than 5 V. The data indicated that this was not a V_{IL} threshold problem for Vendor L's devices but that it might be for Vendor A's.

The same chip in used for the Z8002 microprocessor and the results of this characterization should be indicative of its performance under the same conditions.

Since the characterization software and hardware are in place, additional data, using a larger sample size, should be taken to assess the performance characteristics of other parameters. These include clock thresholds and input/output timing. In addition, the low voltage/high temperature problem should be investigated more thoroughly.

SECTION V

TEST DEVELOPMENT FOR THE Z8000

OBJECTIVE

The purpose of this evaluation was to review the tests which Vendor L submitted for inclusion in the slash sheet for the Z8000. The approach used is defined in "The Procedure for LSI Functional Test Development". It has been documented in the RADC report entitled "Electrical Characterization of Single Chip Microprocessors and Other LSI Devices" and will not be repeated here.

SUMMARY

Vendor L provided an assembly listing of the Z8000 functional test patterns along with a definition of the timing that they use for them. Vendor L performs a dynamic functional test on the Z8000. The timing information was evaluated first since Vendor L indicated that the sequence of vectors in the functional test pattern was subject to change. It was found that not all switching speed parameters were tested during the functional test. Vendor L was contacted and indicated that they were modifying their test program. All parameters would be tested once this revision was completed.

A list of tests required to check the Z8000 was developed and forwarded to Vendor L to assist in the functional test evaluation. A FORTRAN program was also developed to aid in the evaluation which will be completed on another RADC contract.

CIRCUIT DESCRIPTION

The Z8000 CPU, shown in Figure 5-1, is available in four versions. All are sixteen bit, fixed instruction microprocessors fabricated with a high density n channel silicon gate process. They require a single +5 volt power supply and an external clock. The Z8001 and Z8002 operate at a maximum frequency of 4 MHz while the Z8001A and Z8002A operate at 6 MHz.

The segmented Z8001(A) can directly access 8 megabytes of memory and the non-segmented Z8002(A) can access 64 kilobytes. By using the seven segment lines of the Z8001(A), it is possible to divide the eight megabyte address space into one hundred twenty-eight, 64 kilobyte segments. The instruction sets are identical, but due to the larger memory space of the Z8001(A), software compatibility is upward only from the Z8002(A) to the Z8001(A).

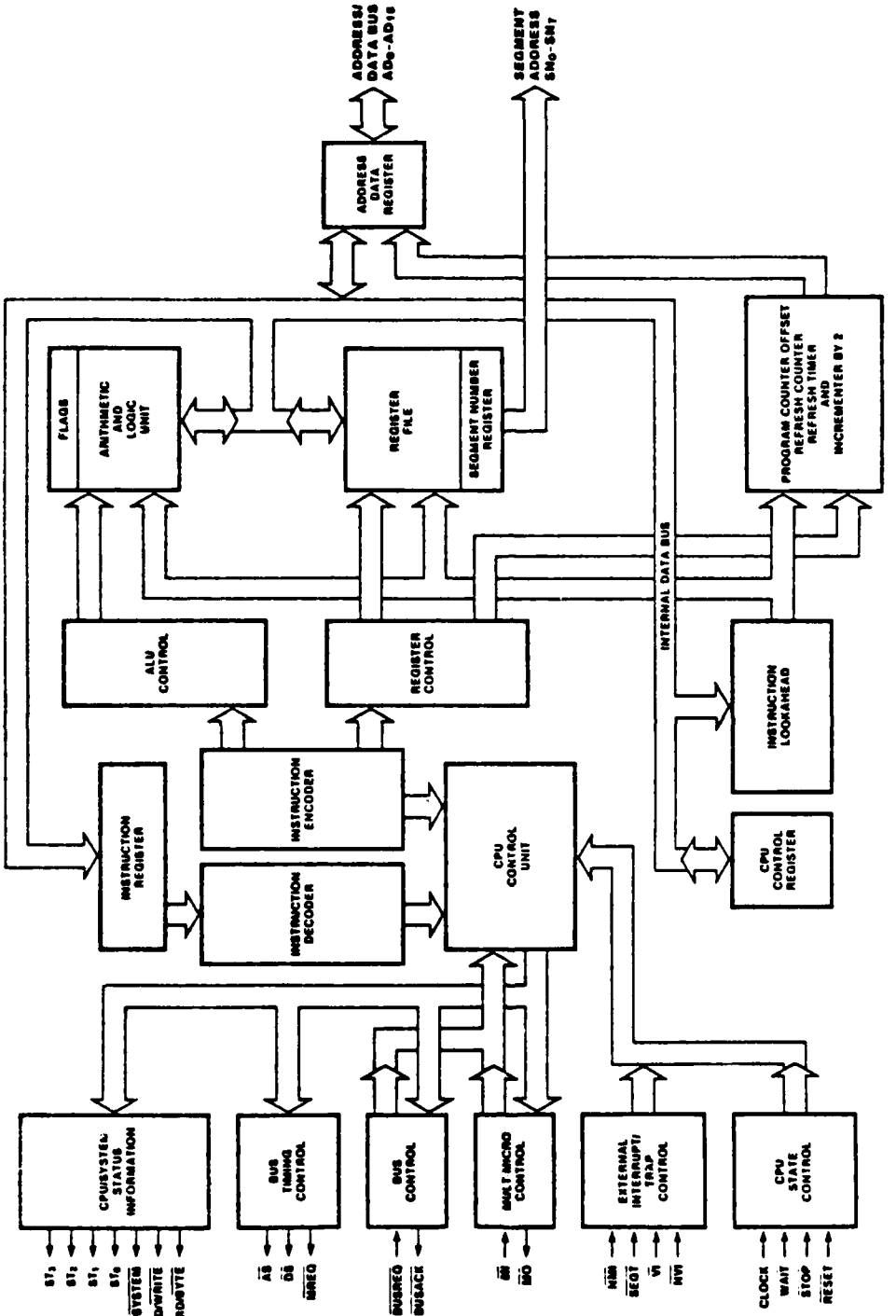


FIGURE 5-1 Z8000 BLOCK DIAGRAM

The microprocessors feature two parallel memory spaces (system and normal) each of which is subdivided into program code, stack, and data spaces. In system mode, the processor can use the entire instruction set while in normal mode it can use only a subset of the instruction set. The I/O space is separate from the memory space and is 64K ports for both devices.

Both microprocessors support multiprocessing with dedicated input and output lines and specialized instructions. The Z8001(A) can be used with a memory management unit which uses a dedicated interrupt input. In addition, there are three interrupt inputs common to both the Z8001(A) and Z8002(A).

The Z8000 architecture features sixteen, 16-bit general purpose registers of which only one has any restrictions on its use as an address component. Eight of the registers can be subdivided to provide sixteen, 8-bit registers. Register concatenation, required for 32 and 64 bit operations, is a predefined function which requires no special commands.

DISCUSSION OF THE FUNCTIONAL TEST EVALUATION PROGRAM

A FORTRAN program was developed to facilitate the evaluation of the functional test vectors. This program creates a record of the register contents on a vector by vector basis. Other records are created for the op codes used, data input or output, and control line inputs and outputs. This information is then used to determine how well the criteria in the Z8000 checklist, listed below, are met.

The program has been tested for all known op codes and conditions in the vendor's literature. A subroutine must be written to input the vectors into the program. This is dependent on the format and media (tape, disk, etc.) in which the vectors are supplied and will be completed when the finalized vector sequence is received from Vendor L. Additional information concerning the organization and use of the program will be included in the report detailing the completed functional test evaluation.

DISCUSSION OF THE FUNCTIONAL TEST

Since the Z8000 is a very complex device, it was sectioned into functional blocks for evaluation. A list of tests required to check each section was developed. This checklist was sent to Vendor L to facilitate the evaluation which will be completed on another RADC contract.

The following is the checklist that was sent to Vendor L.

Z8000 FUNCTIONAL TEST CHECKLIST

1) **ALU and Control Circuits**

   A) **General**

      1) Have all addressing modes of all op codes been tested?

      2) Have all non implemented op codes been tested for proper trapping?

      3) Have all priviledged instructions been attempted under both system and normal operating modes?

      4) Have all "automatic" op codes been tested for correct operation? This includes the verification of the function, modification and verification of address and counter register contents, and termination when the counter register is zero or when the condition code is satisfied.

      5) Have the contents of each register affected by the op code under test been verified prior to further modifications?

   B) Load and Exchange Instructions

      1) Has the clear command been tested for byte and word memory locations and registers and for odd and even addressed bytes and registers?

      2) Has the exchange command been tested for byte and word operands, for odd and even addressed bytes and registers, and by single bit changes?

      3) Has the load command been tested for byte, word, and long word operands and for bytes on odd and even boundaries? What is the effect of addressing a long word load to an odd numbered register "pair"?

      4) Is LDA tested in both the segmented and non-segmented modes? Is LDAR also tested? Has it been verified that the stored address will function with the reserved bits in their undefined state? What is the effect if an odd register "pair" is selected?

      5) Is LDK tested for both four and eight bit constants?

      6) Is LDR tested for byte, word, and long word operands and for operands on odd and even source and destination addresses? What is the effect if a long word load is addressed to an odd register "pair"?

7) Is LDM tested for having loaded the correct number of registers? Is the wrap around of registers greater than sixteen tested?

8) Has it been verified that none of the load instructions affect the flags? This should be done with the flags set at both one and zero.

9) Have the push and pop functions been tested with both word and long word operands? Have all possible registers and pairs been tested for autodecrement (push) and autoincrement (pop) functions? Has the conflict between the immediate push word and the immediate push long word commands been resolved?

10) Has the operation of all of the load and exchange instructions been verified by data reads?

C) Arithmetic

1) Has the add function been tested by adding the four possible bit combinations with and without carry for each bit position? Have the two add instructions been tested for both word and byte operands?

2) Has the subtract function been tested by performing the same tests as for the add function?

3) Has the compare function been tested for byte, word, and long word operands?

4) Has the increment function been tested for both word and byte operands?

5) Has the decrement function been tested for both word and byte operands?

6) Has the multiply function been tested for both data types?

7) Has the divide function been tested for both data types? Has the instruction been aborted by the division by zero, underflow, and overflow conditions? Has the abort been confirmed by both the divide register pair and divide register quadruple variant?

8) Has the sign extension instruction been tested for byte, word, and long word operands?

9) Have all thirteen possible actions of the decimal adjust function been verified? How is the decode circuitry for the decimal adjust implemented? Has it been exercised sufficiently to detect all stuck at faults?

10) Has the flag operation of all of the arithmetic instructions been verified for all possible flag changes?

D) Logical Instruction

1) Have the AND, OR, and XOR instructions been tested by the application of the following patterns to each bit position?

a) (0,0), (0,1), (1,0) for OR operations

b) (0,1), (1,0), (1,1) for AND operations

c) (0,0), (0,1), (1,0), (1,1) for XOR operations

d) Have all three instructions been tested for byte and word operands?

2) Have the complement instructions been tested by complementing each bit for both zero to one and one to zero transitions and for word and byte operands?

3) Has the test instruction been tested for both word and byte operands?

4) Has the correct operation of flags been tested and verified for each logical instruction and for each possible flag combination?

E) Program Control

1) Have all of the program control operations been tested, in both the segmented and non-segmented modes, to insure the correct selection of the implied stack pointer register or register pair?

2) Where used, all condition codes should be tested for correct operation in both the true and false states, while all other flags which are not referenced remain in the opposite state.

a) Is call tested in both segmented and non-segmented modes?

b) Is call relative tested as call and for both positive and negative 2K jumps?

c) Are DJNZ/DBJNZ tested for the full +2, -252 range?

d) Is JP tested for all condition codes and in both segmented and non-segmented modes?

e) Is JPR tested for all condition codes and for the full -254/+256 range?

f) Is RET tested in both segmented and non-segmented modes and for all condition codes?

g) Is system call tested in both segmented and non-segmented modes? In the decrement of the stack pointer register(s) verified?

h) Is IRET tested in both segmented and non-segmented modes? Are the decrement of the stack pointer(s) and disposition of the stack data verified?

F) Bit Manipulation

1) Is BIT tested by setting the selected bit to a true and false state while all other bits are in the opposite state?

2) Are all bits tested and are static and dynamic operations on word and byte data verified?

3) Are bit set and reset instructions tested as in (1) above? Are the test and set instructions tested by setting all bits to zero prior to testing the instruction? Are the proper flags set?

4) Is TCC tested for all possible condition codes, true and false? Is it verified that the indicator bit is set, but not reset, for byte and word operands?

G) Shift and Rotate Instructions

1) Insure left and right rotates function correctly for both single and double position rotations and for byte, word, and long word operands.

2) Insure proper set and reset of carry bit for both single and double position rotations of the rotate through carry commands and for byte and word operands.

3) Insure digit rotates function correctly.

4) Insure correct flag sets/resets for the above commands for all possible combinations of the flag bits.

5) Insure left and right dynamic shifts function correctly.

6) Insure left and right static shifts function correctly.

7) Insure the failure of the instruction if the number of positions to be shifted is greater than that allowed for byte, word, and long word data types. Insure that zero shifts, where undefined, are nondestructive of register data, and that the flags, where defined, are set as required by the data contained in the register(s).

8) Insure that all possible combinations of flag sets/resets are performed as required by the instructions and data.

H) Block Transfer and String Manipulation Instructions

1) Insure that the transfer and string instructions are interruptible and that the instructions continue correctly after an interrupt has been serviced.

2) Compare commands

a) Insure that the commands function for both word and byte operands.

b) Insure that the loop counter register functions properly.

c) Insure that the pointer registers increment/decrement as required by the instruction.

d) Insure that all flags are set and reset as required by the data.

e) Insure that "automatic" functions repeat as required, function as specified, and terminate on both counter at zero and condition code satisfied conditions.

3) Load Commands

a) Insure that the pointer registers are incremented or decremented as required.

b) Insure that the automatic functions repeat as required, function as specified, and terminate on counter register at zero.

c) Insure that the flags are not affected.

4) Translate Test Commands

a) Insure correct translated address output.

b) Insure correct flag setting/resetting.

c) Insure correct pointer register operation.

d) Insure correct termination on either counter register at zero or condition codes non zero as applicable.

I) Input/Output

1) Insure operation of system trap for all instructions when operating in normal mode.

2) Insure proper register modification and termination of the "automatic" input and output instructions. Insure interruptibility and continuation after interrupt of these instructions.

3) What happens when an even byte access is attempted for normal I/O and odd byte addressing is attempted for special I/O? Should these conditions be tested and are they ?

4) Insure that word and byte operands are input or output as required.

5) Insure that the correct flags are set or reset as required.

J) CPU Control Instructions

1) Is COMFLG tested by complementing each of the flags from a 1--0 and 0--1? It should be verified that only one flag is affected at a time. Is the instruction tested for the condition where no flag is defined?

2) Is DI tested by disabling each interrupt separately and then requesting service in both system and normal modes?

3) Is EI tested as in (2) above?

4) Is halt tested for continued memory refresh and recognition of interrupts, reset, and bus requests?

5) Is LDCTL tested for each possible register and are the register contents subsequently verified, in both system and normal modes?

6) Is LDCTLB tested and are the contents of the flag byte verified?

7) Is LDPS tested for segmented and non-segmented operation, in system and normal mode, and are the register contents verified?

8) Are multiprocessor instructions (MBIT, MREQ, MRES, MSET) tested for the proper set or reset of the multimicro out line? Are the correct functioning of the test instructions and flag sets/resets verified? Is register manipulation of the MREQ instruction verified? Are all multiprocessor instructions tested in system and normal modes of operation?

9) Is NOP tested?

10) Are set/reset flag commands tested by changing each selected flag while all other flags remain in their previous state? Is the condition where no flag is defined tested for each command? Are the register contents verified?

3) External Trap/Interrupt Control

A) General

1) Has it been verified that simultaneous traps/interrupts are prioritized as shown in the list below (descending order)

1) reset

2) internal trap

3) non-maskable interrupt

4) segment trap

5) vectored interrupt

6) non-vectored interrupt

2) Has it been verified that reset is serviced regardless of processor state?

3) Has the nesting of interrupts and traps been verified?

4) Has it been verified that the correct PSAP vectors are loaded into the CPU by the interrupt and trap requests? Has it been verified that the correct FCW is loaded?

5) Has the processor's change from normal to system mode been verified for all traps and interrupts?

6) Has the instruction fetch abort been verified for interrupts and traps? Has proper PC operation been verified for this condition?

B) Non Maskable Interrupts

1) Has it been verified that this interrupt is asynchronously detected by activating it at various times?

2) Has it been verified that this interrupt is edge triggered by holding it in the active state and observing that it is serviced just once?

C) Nonvectored Interrupts

1) Insure the mask bit is functional by testing both true and false masks with all other mask bits held in the opposite state.

2) Insure that the vector is correctly translated as a PSAP address pointer.

3) Insure that the input is sampled only during the last clock cycle of an instruction.

4) When will an interrupt be accepted if requested during the first clock cycle of an EI instruction? Is this verified?

D) Vectored Interrupts

1) Insure the mask bit is functional by testing both true and false masks with all other mask bits held in the opposite state.

2) Insure that the vector is correctly translated as a PSAP address pointer.

3) Insure that the input is sampled only during the last clock cycle of an instruction.

4) When will an interrupt be accepted if requested during the first clock cycle of an EI instruction? Is this verified?

E) Notes

1) Each interrupt or trap should be tested with all others inactive, except during the testing for interrupt priority and nesting.

2) The transfer of the correct FCW and PC counter must be verified.

3) The storage in the implied stack of an identifier and the PC and FCW must be verified for all traps and interrupts. This will require the storage of 4 words for the Z8001, but only 3 for the Z8002.

4) Multi Micro Control

A) Is the micro in line received? Is it verified with the micro input test instruction?

5) Bus Control

A) Insure the tri state function of all processor outputs, except $\overline{\text{BUSAK}}$, after the machine cycle in which the $\overline{\text{BUSRQ}}$ was requested.

B) Insure $\overline{\text{BUSAK}}$ is asserted when (A) occurs.

C) Insure return to normal operation 2 clock cycles after $\overline{\text{BUSAK}}$ has been released.

6) CPU Status Information

A) Insure that the status lines can reflect all possible CPU status.

B) Insure that the N/$\overline{\text{S}}$ line reflects the processor control bit.

C) Insure that the R/$\overline{\text{W}}$ line reflects the current operation's function.

D) Insure that the B/$\overline{\text{W}}$ line reflects the current operation's function.

7) Register Control

A) Have all registers (including both sets of stack pointers, the refresh register, the FCW, NPSAP register pair, and the PC register pair) been tested for bit independence by having each bit assume a one and a zero state while all other bits, either individually or collectively, are in the opposite state?

B) Can all word length and byte length registers be addressed/selected? For implied stack operations, is the correct stack register selected by referencing the system/normal mode bit? Has segmented/non-segmented operation been tested?

C) Have the address/data and segment number outputs been tested for bit independence by having each bit assume a one and a zero state while all other bits, either individually or collectively, are in the opposite state?

D) Has it been confirmed that register R0 follows R15 in the multiple register load commands?

E) What are the results of attempting paired and quad register operations on odd boundaries? Is it necessary to attempt these operations? Are they done?

F) Has it been verified that any register may act as a counter (looping commands), stack pointer or index (shift, rotate commands)?

G) Has the segment register been tested for arithmetic isolation from the PC offset? Is this done?

8) CPU State Control

A) Insure refresh operation continues when $\overline{\text{STOP}}$ is applied.

1) Insure one refresh cycle after the release of $\overline{\text{STOP}}$.

2) Insure that $\overline{\text{STOP}}$ is sampled on the falling edge of the clock, preceeding the second word fetched, when the EPU bit is set in the control word and an extended instruction has been fetched.

3) Insure that $\overline{\text{STOP}}$ is sampled on the falling edge of the clock, of the first cycle of an instruction fetch, if the EPU bit is not set.

B) Test reset function to

1) Insure 5 cycle response of processor to $\overline{\text{RESET}}$.

2) Insure that, after $\overline{\text{RESET}}$ is inactive for 3 cycles, the processor fetches 3 words (8001) or 2 words (8002).

3) Insure that the segmented response occurs even when processor was operating in the non-segmented mode.

C) Insure that the WAIT line functions

9) Miscellaneous

A) High Impedance Capability

1) Insure that AD0-AD15, $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{MREQ}}$, R/$\overline{\text{W}}$, B/$\overline{\text{W}}$, ST0-ST3, N/$\overline{\text{S}}$, SN0-SN6 enter the high Z state as required by the bus request/acknowlege sequence.

2) Insure AD0-AD15 enter high Z state when internal cycles occur and during wait states.

B) Insure the function of the refresh circuits

1) Masked and non-masked operation.

2) Insure multiply instruction functions with refresh running.

3) Test all possible refresh ratios and verify their correct operation.

4) Insure that refresh is accomplished before an interrupt or trap is honored (simultaneous arrival of requests).

5) Insure that the auto refresh of skipped refresh address occurs after a skipped refresh period, along with the normally occurring refresh.

C) Other architectural considerations

1) Is the program counter incremented by the ALU or separate circuitry? If a separate circuit is used, it must be verified that it functions as specified. This should include testing which verifies the change of state of the most significant as well as low order bits. Isolation of the offset value from the segment value should also be verified, using the overflow of the offset register as the test vehicle.

2) Is a separate arithmetic unit used to increment or decrement the address registers or to decrement the counter register in the auto increment/decrement instructions? If so, this unit should be tested as in (1) above.

3) Several constant values are supplied to the ALU. These include the decimal adjust command's correction values, the incremental or decremental values for the auto increment and decrement instructions, etc. All constants should be verified during the functional test.

APPENDIX A

Z8001 TASK BENCHMARKS

1. Z8001 Block Move

This routine moves a block of data from one point in memory to any other point in memory.

Register Use
- RR2 Address of the first word of the source block
- RR4 Address of the first word of the destination block
- R6 Number of words to be moved

Symbol Table
- Stara: Address of first word, source block
- Stard: Address of first word, destination block
- Count: Number of words to be moved

Command	Bytes	Cycles	Comment
LDL RR2, #Stara	6	11	Initialize register
LDL RR4, #Stard	6	11	Initialize register
LD R6, #Count	4	7	Initialize register
LDIR RR4, RR2, R6	4	11 + 9n	Perform move and loop

Total bytes: 20

Total cycles: 40 + 9n

where n is the number of array elements

2. Z8001 Array Addition

This routine adds two arrays of equal arbitrary length, composed of 16 bit words, located anywhere in memory. The result, also assumed to be 16 bits, is then stored in any desired memory location.

Register Use

R0	Result of addition
R1	Offset of all three terms
R2	Number of elements to be added
RR4	Base register for A array
RR6	Base register for B array
RR8	Base register for result array

Symbol Table

A:	Offset value for A array
B:	Offset value for B array
C:	Offset value for result array
Count:	Number of elements to be added

	Command	Bytes	Cycles	Comments
	LDL RR4, A	6	11	Initialize registers
	LDL RR6, B	6	11	Initialize registers
	LDL RR8, C	6	11	Initialize registers
	CLR R1	2	7	Initialize registers
	LD R2, #Count	4	7	Initialize registers
Loop	LD R0, R1 (RR4)	4	14	First word loaded
	ADD R0, (RR6)	2	7	Second word added
	LD R0 (RR8), R0	4	14	Result to desired array
	INC R1, 2	2	4	Index register updated
	INC R7, 2	2	4	B array pointer updated
	DJNZ R2, Loop	2	11	Loop if not all added

Total bytes: 40

Total cycles: 47 + 54n

where n is the number of array elements

3. Z8001 Array Multiplication

This routine multiplies two arrays of equal arbitrary length, composed of 16 bit words, located anywhere in memory. The 32 bit result is then stored in any desired memory location.

Register Use	RR2	Results of multiplication
	R4	Number of elements to be multiplied
	RR6	Base address of multiplier array
	RR8	Base address of multiplicand array
	RR10	Base address of result array

Symbol Table	A:	Base address of multiplicand
	B:	Base address of multiplier
	C:	Base address of product array
	Count:	Number of elements to be multiplied

	Command	Bytes	Cycles	Comment
	LDL RR6, #A	6	11	Initialize register
	LDL RR8, #B	6	11	Initialize register
	LDL RR10, #C	6	11	Initialize register
	LD R4, #Count	4	7	Initialize register
Loop	LD R3, (RR6)	2	7	Load multiplicand
	MULT RR2, (RR8)	2	70	Perform multiplication
	LDL (RR10), RR2	2	17	Store result
	INC R7, 2	2	4	Increment address pointer
	INC R9, 2	2	4	Increment address pointer
	INC R11, 4	2	4	Increment address pointer
	DJNZ R4, Loop	2	11	Decrement element count and loop

Total bytes: 36

Total cycles: 40 + 117n

where n is the number of array elements

4. Z8001 Array Sort

This routine sorts an arbitrary length array of 16 bit words into descending order. The resulting array may be located anywhere in memory, while the source array is destroyed.

Register Use

R0	Largest value found
R3	Working counter
R4	Number of words to be sorted
R5	Working counter
RR8	Source array base address
RR10	Destination array base address
RR12	Working address in source array
RR14	Address of largest word

Symbol Table

A:	Unsorted array base address
B:	Sorted array base address
Count:	Number of words to be sorted

		Command	Bytes	Cycles	Comment
1.		LD R4, #Count	4	7	Initialize registers
2.		LD R3, R4	2	3	Initialize registers
3.		LD 1 RR8, #A	6	11	Initialize registers
4.		LD 1 RR10, #B	6	11	Initialize registers
5.	Oloop	LD R5, R4	2	3	Initialize inner loop
6.		LD 1 RR2, RR8	2	5	Initialize inner loop
7.	Iloop	LD R0 (RR2)	2	7	Update largest word
8.		LDL RR14, RR12	2	5	Save location of that word
9.		CPIR R0, RR12, R5, GT	2	11 + 9n	Compare to string
10.		JP 7, Iloop	4	8	Update if larger
11.		LD (RR10), R0	2	8	Transfer to sorted array
12.		CLR (RR14)	2	7	
13.		INC R11, #2	2	4	Increment pointer
14.		DEC R3, #1	2	4	Decrement outside counter
15.		DJNZ R4 Oloop	6	10	Jump if not all sorted

Total bytes: 46

Worst case cycles $9n^2 + 72n + 32 + 12(1+2+3...(n-1))$

Best case cycles $9n^2 + 72n + 32$

Where n is the number of array elements

Timing notes

Worst case timing, when the array is in ascending order, will require that lines 7 and 8 be repeated (n - (D - 1)) times, where D is the number of words already sorted. The best case timing, with the array already in ascending order, will not repeat these lines at all.

5. Z8001 Interrupt Service

This routine services an interrupt storing 8 word length registers and restoring them before returning to the main routine. The latency time is that of a register word length division, which is comparable to the 8086 IDIV instruction

Action	Cycles	Comment
Latency	107	DIV, Register
Abort following instruction fetch	3	
Interrupt acknowledge	8	
Save 4 word registers	36	Push FSW and PC
Get new processor status	12	Load new FSW and PC
Save 8 word registers	72	Push
Restore 8 word registers	64	Pop
Return from Interrupt	16	IRET

Total cycles = 318

Time required = .08 msec

APPENDIX B

8086 TASK BENCHMARKS

1. 8086 Block Move

This routine moves a block of data from one point in memory to any other point in memory

Register Use
- SI Offset of first word of destination array
- DI Offset of first word of source array
- ES Segment where destination string is to be located
- DS Segment where source string is located
- CX Number of words to be moved

Symbol Table
- Stara: Offset of first word of source array
- Stard: Offset of first word's location in destination array
- DSEG: Destination segment
- SSEG: Source string segment location
- Count: Number of words to be moved

Command	Bytes	Cycles	Comment
CLD	1	2	Clear direction flag
MOV DI, #Stara	3	4	Destination index set
MOV SI, #Stard	3	4	Source index set
MOV AX, #DSEG	3	4	Destination segment to accumulator
MOV ES, AX	2	2	Destination segment set
MOV AX, #SSEG	3	4	Source segment to accumulator
MOV DS, AX	2	2	Source segment set
MOV CX, #Count	3	4	Count register initialized
REP NZ MOVS	2	9 + 17n	Perform moves

Total bytes: 22

Total cycles: 35 + 17n

where n is the number of array elements

2. 8086 Array Addition

This routine adds two arrays of equal arbitrary length, composed of 16 bit words, located anywhere in memory. The result, also assumed to be 16 bits, is then stored at any desired memory location.

Register Use
- BX Offset value for added array
- DS Location segment of "A" array
- ES Location segment of "B" array
- SS Location segment of result array
- BP Offset value for result array

Symbol Table
- Stara: Constant offset of "A" array
- Stard: Constant offset of "B" array
- Starr: Initial offset of result array
- Count: Number of elements to be added
- Seg D: Segment where "A" array is located
- Seg E: Segment where "B" array is located
- Seg S: Segment where result is to be located

	Command	Bytes	Cycles	Comment
	MOV BX, #0	3	4	Initialize registers
	MOV AX, #Seg S	3	4	Initialize registers
	MOV SS, AX	2	2	Initialize registers
	MOV BP, #Starr	3	4	Initialize registers
	MOV CX, #Count	3	4	Initialize registers
	MOV AX, #Seg D	3	4	Initialize registers
	MOV DS, AX	2	2	Initialize registers
	MOV AX, #Seg E	3	4	Initialize registers
	MOV ES, AX	2	2	Initialize registers
Loop	MOV AX, Stara (BX)	4	18	Move first word to accumulator
	ADD AX, ES: Stard (BX)	4	20	Add second result to accumulator
	MOV (BP), AX	3	13	Move result to storage array
	ADD BP, #2	4	4	Update address pointer
	ADD BX, #2	4	4	Update address pointer
	LOOP NZ Loop	2	19/6	Loop to add if CX = 0

Total bytes: 45

Total cycles: 36 + 78n

where n is the number of array elements

3. 8086 Array Multiplication

This routine multiplies two arrays of equal arbitrary length, composed of 16 bit words, located anywhere in memory. The 32 bit result is then stored at any desired location in memory.

Register Use
- BX Offset of operands
- BP Offset of product
- DS Segment where multiplicand is located
- ES Segment where multiplier is located
- SS Segment where product array is to be stored
- CX Number of elements to be multiplied

Symbol Table
- Star A: Constant offset of multiplicand array
- Star B: Constant offset of multiplier array
- Star C: Initial offset address of product array
- Seg S: Segment where result array is to be stored
- Seg D: Segment where multiplicand array is stored
- Seg E: Segment where multiplier array is stored
- Count: Number of elements to be multiplied

	Command	Bytes	Cycles	Comment
	MOV BX, #0	3	4	Set multiplier element offset
	MOV BP, #Star C	3	4	Result offset initialized
	MOV AX, #Seg D	3	4	Multiplicand segment to accumulator
	MOV DS, AX	2	2	Multiplicand segment initialized
	MOV AX #Seg E	3	4	Multiplier segment to accumulator
	MOV ES, AX	2	2	Multiplier segment initialized
	MOV AX, #Seg S	3	4	Result segment to accumulator
	MOV SS, AX	2	2	Result segment initialized
	MOV CX, #Count	3	4	Element count initialized
Loop	MOV AX, ES: Star A (BX)	5	19	Multiplicand to accumulator
	IMUL Star B (BX)	4	149	Multiplication done
	MOV Star C (BP), AX	3	14	Store high element
	ADD BP, #2	4	3	Increment result pointer
	MOV Star C (BP), AX	3	14	Stored on element
	ADD BP, #2	4	3	Increment result pointer
	ADD BX, #2	4	3	Increment operand pointer
	LOOP NZ, Loop	2	1/6	Loop until CX = 0

Total bytes: 51

Total cycles: 36 + 230n

where n is the number of array [illegible]

4. 8086 Array Sort

This routine sorts an array of words located anywhere in memory into descending order. The resulting array is stored in any other location in memory, while the source array is destroyed.

Register Use

BP	Offset pointing to storage location of next sorted word
CX	Internal loop counter
DX	External loop counter
DI	Internal loop address index 1
SI	Internal loop address index 2
ES	Number of words to be sorted
DS	Segment where unsorted array is stored
SS	Segment where sorted array is to be stored

Symbol Table

Count:	Number of words to be sorted
Seg D:	Unsorted array's segment
Star S:	Unsorted array's offset
Seg S:	Sorted array's segment
Star D:	Initial word's offset, sorted array

		Command	Bytes	Cycles	Comment
1.		MOV AX, #Seg S	3	4	Sorted segment initialized
2.		MOV SS, AX	2	2	Sorted segment initialized
3.		MOV AX, #Count	3	4	Count initialized
4.		MOV DX, AX	2	2	Count initialized
5.		DEC AX	1	2	Count initialized
6.		MOV ES, AX	2	2	Count initialized
7.		MOV AX, #Seg D	3	4	Unsorted segment initialized
8.		MOV DX, AX	2	2	Unsorted segment initialized
9.		MOV BP, #Star D	3	4	Sorted offset initialized
10.	Oloop	MOV CX, ES	2	2	Internal count initialized
11.		MOV SI, #0	3	4	Loop index initialized
12.		MOV DI, #2	3	4	Loop index initialized
13.		MOV AX, Star S (SI)	4	17	Load first word
14.	Iloop	CMP AX, Star S (DI)	4	18	Compare to second
15.		JLE Cont	2	16/4	Jump if first is greater
16.		MOV SI, DI	2	2	Move element address
17.		MOV AX, Star S (DI)	4	17	Move larger element
18.	Cont	ADD DI, #2	3	4	Increment index
19.		Loop NZ Iloop	2	19/5	Loop until all are compared
20.		MOV Star D, (BP)	3	18	Move largest to sorted array
21.		CMP DX, #0	3	4	All elements sorted?
22.		JZ out	2	16/4	If so, done
23.		MOV Star S (SI), #0	6	19	Clear largest value
24.		ADD BP #2	3	4	Increment storage pointer
25.		DEC DX	1	2	Decrement outer counter
26.		JMP Oloop	2	15	Begin sort again
27.	Out	Halt			

Total bytes: 70

Worst case cycles: $91n^2 - 20n + 41$

Best case cycles: $84n^2 - 13n + 41$

where n is the number of array elements

Timing notes

a) The compare loop (lines 10 - 19) is done (n-1)n times. Thus lines 10, 11, 12, 13, 14, 18 and 19 (taken) occur n(n-1) times per use of the routine.

b) The storage of the largest element found in the unsorted array occurs n times per routine use. Thus lines 19 (not taken), 20, 21, 22 (not taken) 23, 24, 25 and 26 occur n times.

c) Lines 1 through 9 and 22 (taken) can be considered overhead, and occur once per routine use.

d) In the best case situation, the routine is already in descending order. Thus the jump on line 15 is taken n(n-1) times and lines 16 and 17 are not used.

e) In the worst case situation, with the array to be sorted in ascending order, the jump on line 15 is never taken, so lines 16, 17, and 15 (not taken) are used n(n-1) times.

Best case = a + b + c + d

= 68(n)(n-1) + 71n + 41 + 16(n)(n-1)

= $84n^2 - 13n + 41$

Worst case = a + b + c + e

= 68(n)(n-1) + 71n + 41 + 23(n)(n-1)

= $91n^2 - 20n + 41$

5. 8086 Interrupt Service

This routine services an interrupt sorting 8 word length registers and restoring them before returning to the main routine. The latency time is that of the IDIV instruction, which is comparable to the Z8001 word length division

Action	Cycles	Comment
Latency	184	IDIV
Interrupt Processing	61	Given by manufacturer
Store 8 word registers	88	
Restore 8 word registers	64	
Return from interrupt	24	

Total cycles = 421

Calculated time = .08 msec

Expected run time (+5%)= .09 msec

APPENDIX C

ARITHMETIC MIX COMPOSITION

This mix is intended to test the relative arithmetic efficiency of microprocessors. It is based on the mix "N", fire control simulation mix, but lacks the floating point and transcendental operations of that mix, as these operations are not hardware supported by either processor. The 23.2% of the total instruction mix which was excluded has been proportionally divided among the remaining categories of instructions. The input/output operations were specifically increased in importance, however, to test the specific I/O commands available to the processors

Addressing modes are also tested beyond mix "N" specifications, as set out in the table below. The base address category will, however, not only include base address modes, but other addressing modes not included under the direct address, indexed, or indirect register modes. These modes are listed under the specific operation.

All operations are assumed to be on 16 bit, binary numbers, unless otherwise mentioned. Signed numbers are also assumed.

Operation	Mix "N" less floating point operations	Microprocessor Mix used for benchmark
1) Data Transfers	46	59.1
2) Arithmetic		
Addition	3	3.95
Subtraction	3	3.95
Multiplication	.15	.2
Division	.05	.1
3) Shift/Rotate	1.5	1.95
4) Compare	.7	.95
5) Branches		
Unconditional	3.1	4.0
Conditional, taken	3.1	4.0
Conditional, not taken	3.8	4.95
Loop control, taken	3.8	4.95
Loop control, not taken	1.5	1.95
6) Index Register Operations	5.5	7.21
7) Logical	1.5	1.95
8) Input/Output	.1	.79
Totals	76.8%	100%

Addressing Modes

50% of addressing is in direct mode
25% of addressing is in the indirect mode
15% of addressing is in the indexed mode
10% of addressing is in the base mode

APPENDIX D

Z8001 ARITHMETIC MIX RESULTS

		Cycles	Totals
1.	Data transfers — 5910 operations		
a)	2955 register to memory		
1)	1478 direct address		
	739 short offset	12	8,869
	739 long offset	14	10,346
2)	739 indirect register	8	5,904
3)	443 indexed address		
	222 short offset	12	2,264
	221 long offset	15	3,315
4)	295 based (BX, BA, RA)	14	4,130
b)	2955 memory to register		
1)	1478 direct address		
	739 short offset	10	7,390
	739 long offset	12	8,868
2)	739 indirect address	7	5,173
3)	443 indexed address		
	222 short offset	10	2,220
	221 long offset	13	2,873
4)	295 based		
	199 BA, BX, or RA	14	2,756
	96 immediate	7	672
	Total cycles required: 64,780		
2.	Arithmetic — 820 operations		
a)	Addition 395 operations		
1)	198 direct address		
	99 short offset	10	990
	99 long offset	12	1,188
2)	99 indirect register	7	693
3)	59 indexed		
	30 short offset	10	300
	29 long offset	13	377
4)	39 based		
	20 register	4	80
	19 immediate	7	133

2. Arithmetic (cont'd.)

	Cycles	Totals
b) Subtraction 395 operations		
1) 198 direct address		
99 short offset	10	990
99 long offset	12	1,188
2) 99 indirect register	7	693
3) 59 indexed		
30 short offset	10	300
29 long offset	13	377
4) 39 based		
20 register	4	80
19 immediate	7	133
c) Multiplication 20 operations		
1) 10 direct address		
5 short offset	72	360
5 long offset	74	370
2) 5 indirect register	70	350
3) 3 indexed		
2 short offset	72	144
1 long offset	75	75
4) 2 based (immediate or register)	70	140
d) Division 10 operations		
1) 5 direct address		
3 short offset	97	291
2 long offset	99	198
2) 2 indirect register	95	190
3) 2 indexed address		
1 short offset	97	97
1 long offset	100	100
4) 1 based (immediate or register)	95	95

Total cycles required: 9932

3. Shift/rotate 195 operations

a) Rotate 98 operations		
49 rotate right/left 1 place	6	294
49 rotate right/left 2 places	7	343
b) Shift 97 operations		
49 shift right/left 1 place	18	882
48 shift right/left 8 places	39	1,872

Total cycles required: 3191

4. Compare — 95 operations

	Cycles	Totals
a) 43 direct address		
22 short offset	10	220
21 long offset	12	252
b) 24 indirect register	7	168
c) 17 indexed		
9 short offset	10	90
8 long offset	13	104
d) 11 based		
6 immediate	7	42
5 register	4	20

Total cycles required: 896

5. Branch instructions — 1985 total operations

a) Branch instructions 1295 operations		
1) 648 direct address		
324 short offset	8	2,592
324 long offset	10	3,240
2) 324 indirect register		
216 unconditional or taken	15	3,240
108 not taken	10	1,080
3) 194 indexed address		
97 short offset	8	776
97 long offset	11	1,067
4) 129 base addressed (RA)	6	774
b) Loop control 690 operations	11	7,590

Total cycles required: 20,359

6. Index register operations — 721 operations

1) 361 direct address		2353
181 short offset	13	637
180 long offset	15	735
2) Indirect register mode not available for this instruction		2700
3) 180 indexed address		1170
90 short offset	13	325
90 long offset	16	1440
4) 180 base address (BX, BA, RA)		
48 no timing differences	15	2700

Total cycles required: 10363

7. Logical operations	195 operations	
	Cycles	Totals
a) AND 65 operations		
1) 33 direct address		
17 short offset	10	170
16 long offset	12	192
2) 16 indirect register	7	112
3) 10 indexed		
5 short offset	10	50
5 long offset	13	65
4) 6 based addressed		
3 immediate	7	21
3 register	4	12
b) Complement 65 operations		
1) 33 direct address		
17 short offset	16	272
16 long offset	18	288
2) 16 indirect register	12	192
3) 10 indexed register		
5 short offset	16	80
5 long offset	19	95
4) 6 based address		
6 register	5	30
c) OR 65 operations		
1) 33 direct address		
17 short offset	10	170
16 long offset	12	192
2) 16 indirect address	7	112
3) 10 indexed address		
5 short offset	10	50
5 long offset	13	65
4) 6 based		
3 immediate	7	21
3 register	4	12

Total cycles required: 2201

8. Input/Output operations 79		
40 input/output, direct address	12	480
39 input/output, indirect address	10	390

Total cycles required: 870

Cycles required for entire mix: 112,592

Time required to "execute" mix: 28.2 msec

APPENDIX E

8086 ARITHMETIC MIX RESULTS

The following address modes are used in both arithmetic and word processing mix evaluations:

Mix Mode	8086 Address Mode
Direct Address	Direct 16 bit offset added to D.S. This takes 6 clock cycles to compute.
Indirect Register Address	Indirect, through the sum of a base or index register added to D.S. The calculation requires five clock cycles.
Indexed Address	Indirect, through the sum of a base register and an index register added to D.S. Computation of the address requires 8 clock cycles.
Base Indexed Address	The same as indexed, but with a constant offset added. This address requires 12 clock cycles to compute.
Base Address	Indirect, through the sum of a base or index register, a displacement constant, and D.S. This address requires 9 clock cycles to compute.

1. Data transfer operations — 5910 operations

	Cycles	Total
a. 2955 Register to Memory		
1) 1478 Direct address	15	22,170
2) 739 Indirect register	14	10,346
3) 443 Indexed	17	7,531
4) 295 Based		
148 Based	18	2,664
147 Base Indexed	21	3,087

1. Data transfer operations (cont'd) 5910 operations

	Cycles	Total
b. 2955 Memory to register		
1) 1478 Direct address	14	20,692
2) 739 Indirect address	13	9,607
3) 443 Indexed	16	7,088
4) 295 Based		
99 Immediate	4	396
93 Base	17	1,666
98 Base indexed	20	1,960

Total cycles required: 87,207

2. Arithmetic 820 operations

a. Addition 395 operations		
1) 198 Direct address	15	2,970
2) 99 Indirect address	14	1,386
3) 59 Indexed	17	1,003
4) 39 Based		
10 Register	3	30
10 Immediate	4	40
10 Base	18	180
9 Base Indexed	21	189
b. Subtraction 395 operations		
1) 198 Direct address	15	2,970
2) 99 Indirect address	14	1,386
3) 59 Indexed	17	1,003
4) 39 Based		
10 Register	3	30
10 Immediate	4	40
10 Base	18	180
9 Base Index	21	189
c. Multiplication 20 operations		
1) 10 Direct address	150	1,500
2) 5 Indirect address	149	745
3) 3 Indexed address	152	456
4) 2 Based		
1 Base	153	153
1 Index base	156	156

2. Arithmetic (cont'd.)

	Cycles	Total
d. Division 10 operations (16 bit signed)		
1) 5 Direct address	183	915
2) 2 Indirect register	182	364
3) 2 Indexed	185	370
4) 1 Base addressed	186	186

Total cycles required: 16,441

3. Shift/Rotate — 195 operations

a. Rotate 98 operations		
1) 49 Rotate Right/left 1 bit	2	98
2) 49 Rotate Right/left 2 bits	16	784
b. Shift 97 operations		
1) 49 Shift Right/left 1 bit	2	98
2) 48 Shift Right/left 8 bits	40	1,920

Total cycles required: 2,900

4. Compare — 95 operations

a. 43 Direct address	15	645
b. 24 Indirect address	14	336
c. 17 Indexed	17	289
d. 11 Based		
Register 3	3	9
Immediate 3	4	12
Based 3	18	54
Base indexed 2	21	42

Total cycles required: 1387

		Cycles	Total
5.	Branch Instructions	1985 total operations	
	a. 1295 Branch instructions		
	1) 648 Direct address		
	324 conditional taken	16	5,184
	324 conditional not taken	4	1,296
	2) 324 Indirect register (JP)	11	3,564
	3) 194 Indexed (JP)	26	5,044
	4) 129 Based		
	Inter segment based (33)	33	1,089
	Inter segment base indexed (32)	36	1,152
	Intra segment based (32)	24	768
	Intra segment base indexed (32)	27	864
	b. Loop control instructions 690 operations		
	Loop, taken (172)	17	2,924
	Loop, not taken (173)	5	865
	Loops taken (172)	19	3,268
	Loops not taken (173)	5	865
	Total cycles required: 26,883		
6.	Index Register Operations	721 operations (LEA)	
	a. 361 Direct address	8	2888
	b. 180 Indirect address	7	1260
	c. 108 Indexed	10	1080
	d. 72 Base		
	36 Base	11	396
	36 Based indexed	14	504
	Total cycles required: 6128		

		Cycles	Total
7.	Logical operations	195 operations	
a)	AND 65 operations		
1)	33 direct address	15	495
2)	16 indirect register	14	224
3)	10 indexed	17	170
4)	6 based		
	register 2	3	6
	immediate 2	4	8
	based 1	18	18
	base indexed 1	21	21
b)	Complement 65 operations		
1)	33 direct address	22	726
2)	16 indirect register	21	336
3)	10 indexed	24	240
4)	6 based		
	register 2	3	6
	base indexed 2	28	56
	based 2	25	50
c)	OR 65 operations		
1)	33 direct access	15	495
2)	16 indirect register	14	224
3)	10 indexed	17	170
4)	6 based		
	register 2	3	6
	immediate 2	4	8
	based 1	18	18
	base indexed 1	21	21
	Total cycles required: 3,298		
8.	Input/Output	79 operations	
40	Input/Output fixed port	10	400
39	Input/Output variable port	8	312
	Total cycles required: 712		

Total cycles calculated to be required: 144,591

Calculated time required at 5 MHz: 29.0 msec

Expected "run" time (5% variance): 30.0 msec

APPENDIX F

CRT TERMINAL CONTROLLER MIX DEFINITION

This section details assumptions made about the specific task and assumptions made about the system architecture from which the CRT Terminal Controller mix was extracted.

Task Definition

The task is based on a microprocessor controlled, stand alone terminal of the "electronic typewriter" type. All operations are concerned exclusively with the input, modification, or output of ASCII characters.

The task itself consists of the input, storage and display of 3 pages of 48 lines with 80 characters per line. Operator errors are assumed to occur at a rate of 5 errors per 80 characters written. Two of these require the deletion of a character, two require the insertion of a character, and one the changing of a character. After the entire three pages have been written, line 23 of the second page will be deleted and a new one inserted in its place. The text is scrolled from top to bottom once and then output to the printer.

Although the microprocessor could be performing other tasks at the same time, only the instructions necessary to accomplish the above operations have been included in the benchmark. Furthermore, ideal system response has been assumed, since this is to be a comparison of microprocessor efficiencies rather than hypothetical system design. Some of the terminology of the basic CRT based machine is included in the flow charting, because it is descriptive and easy to use. It does not affect the mix.

System Assumptions

The system, shown in Figure F-1, consists of two major components, the control unit and the display unit.

The control unit contains the microprocessor and its supporting circuits, the system memory, and the keyboard and printer buffers. Some type of interrupt masking capability is assumed, either on the microprocessor or as a supporting peripheral. The system memory is accessible to the microprocessor at all times.

The display unit contains a video timer and controller (VTAC), a 4K X 8 (48 line) display buffer memory, an offset latch for display memory addressing, supporting circuits for the VTAC, and address decoding logic. The VTAC references the display memory for the display updates, allowing the microprocessor access to the display memory during vertical blanking intervals.

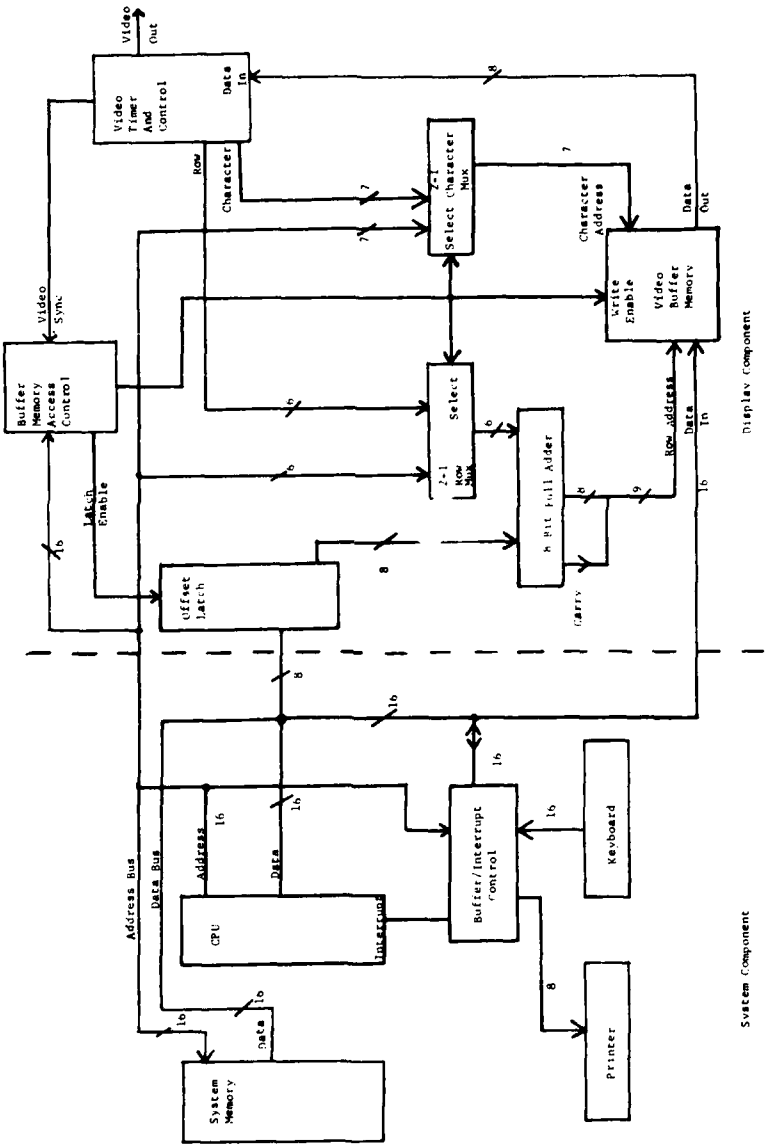


FIGURE F-1 SYSTEM BLOCK DIAGRAM

It should be noted that the cursor address and microprocessor video memory address registers do not necessarily point to the actual memory row address. The offset latch must be added to them to generate the actual row address. This latch is accessible under the same conditions as the video unit memory.

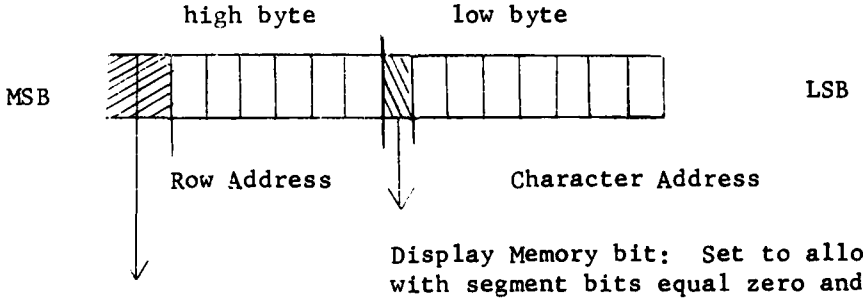
Register Usage

Due to the limited nature of the task, the registers used are maintained intact throughout the overall routine. Five registers are dedicated to the storage or manipulation of data as defined below.

Register	Function
"A"	System memory address register
"B"	Video memory address
"C" high byte	Offset latch value
"C" low byte	Page or half page boundary for system's displayed row Ø
"E"	Multiple use (scratch pad) register
"F"	Multiple use (scratch pad) register
"G" high byte	Maximum segment and Row used (system memory)
"G" low byte	Multiple use (scratch pad) register

Address Word Definition

The address bus, which is assumed to be 16 bits, is partitioned as below.



Display Memory bit: Set to allow access, with segment bits equal zero and video interrupt set, to allow access to the display buffer memory.

Segment bits: In conjunction with bit 7, these bits allow access to 4 system pages of 48 lines plus the 48 lines of display memory.

Access to the video buffer memory requires that both the segment bits and display memory bit be set to zero.

With the display memory bit set to one , the segment bits allow access to four pages of system memory. Data is stored in an x-y addressable block to reduce the time required for system memory to video buffer memory data transfers.

Flow Chart Language Notes

To decrease the effects of bias and experience on the benchmarking process, the flow charts for the required processor routines are written in a hybrid language. Notes on this language are given below.

Addressing Mode	Chart Symbolic	Definition
Indirect Register	@ "X"	Register contains address of operand
Register	"X"	Register holds operand
Immediate	None	Operand is part of, or immediately follows instruction

General

a) Instruction format is

Instruction source, destination

b) Instructions such as set/clear bit are given in English for clarity, although they are translated to assembly language for mix instruction set extraction.

c) Hexidecimal numbers are used exclusively in the flow charts.

Flow Charts and Descriptions

The following pages contain flow charts and descriptions of the various routines that are used.

Character Write Routine

This routine separates keyboard inputs into two separate groups. Control inputs are then directed to the appropriate subroutines, while character inputs are stored in both the display buffer and system memory. The final section of the routine updates the address pointers and returns control to the "wait" loop.

Section A performs general housekeeping and insures that no further keyboard interrupts will be honored until the initial interrupt has been serviced. An interrupt from the video unit is allowed and the routine loops until access to the video buffer memory is allowed.

Section B separates allowed control functions from character inputs.

Section C stores the character.

Section D updates (increments) the cursor character position, the simplest case of updating this address.

Section E is reached when the cursor points to the final character address of a row. The row is incremented (if possible), the cursor and memory pointers are set to the first character of the new row, and control is returned to the "wait" loop.

Section F resolves the case where the maximum number of rows addressable by the VTAC has been reached, but the offset buffer is not filled to capacity. The buffer is incremented, the pointer set to zero, and the program returns to the "wait" loop.

Section G is a parameter initialization segment, since this portion of the routine is needed only when both the number of rows and the offset buffer are at maximum capacity. This implies the need to update the entire video buffer and, with the registers set to the required values, the display update routine is called to perform this task.

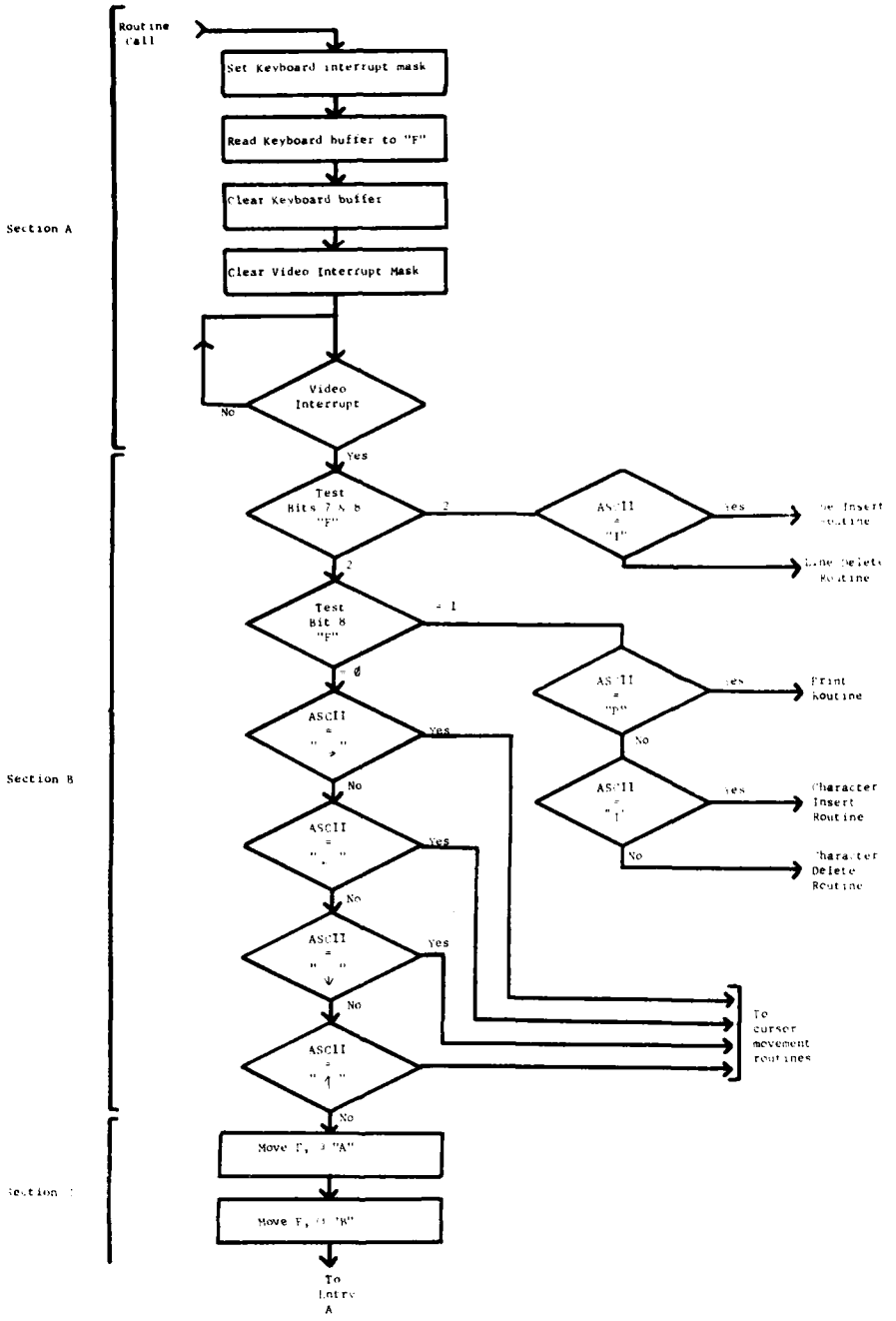


FIGURE F-2A CHARACTER WRITE ROUTINE

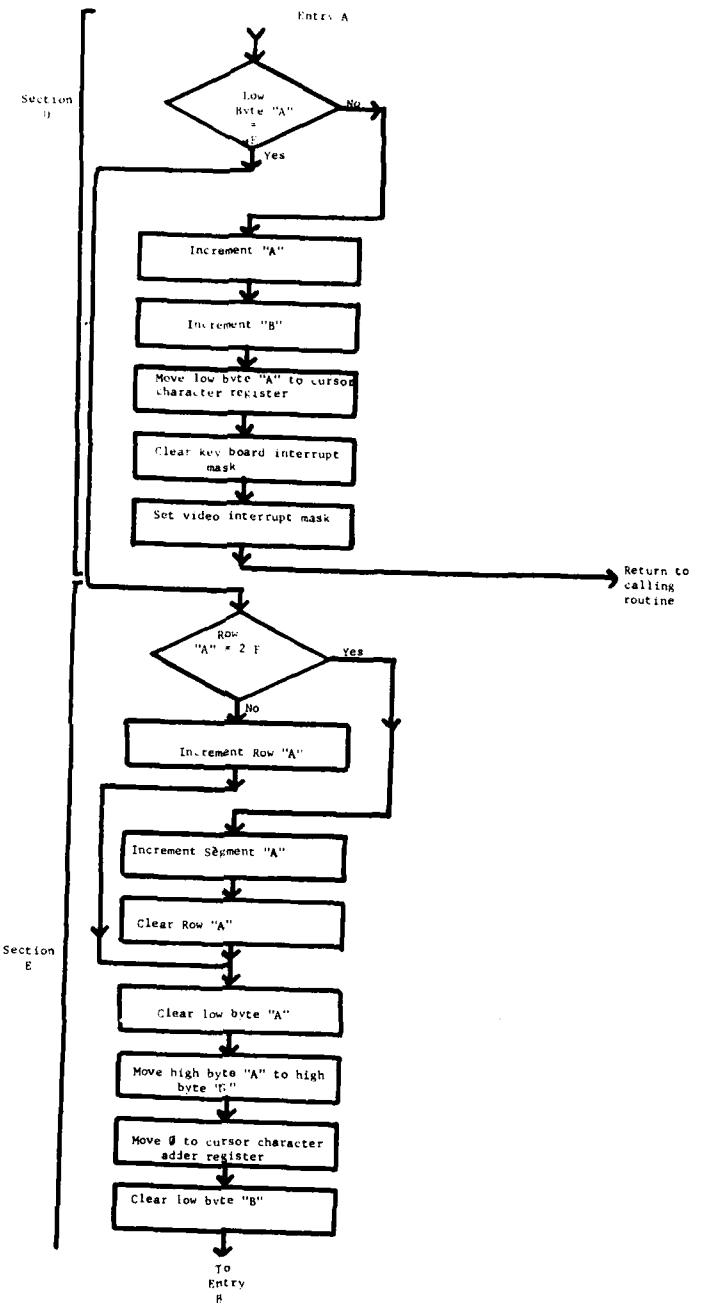


FIGURE F-2B CHARACTER WRITE ROUTINE, CONT'D

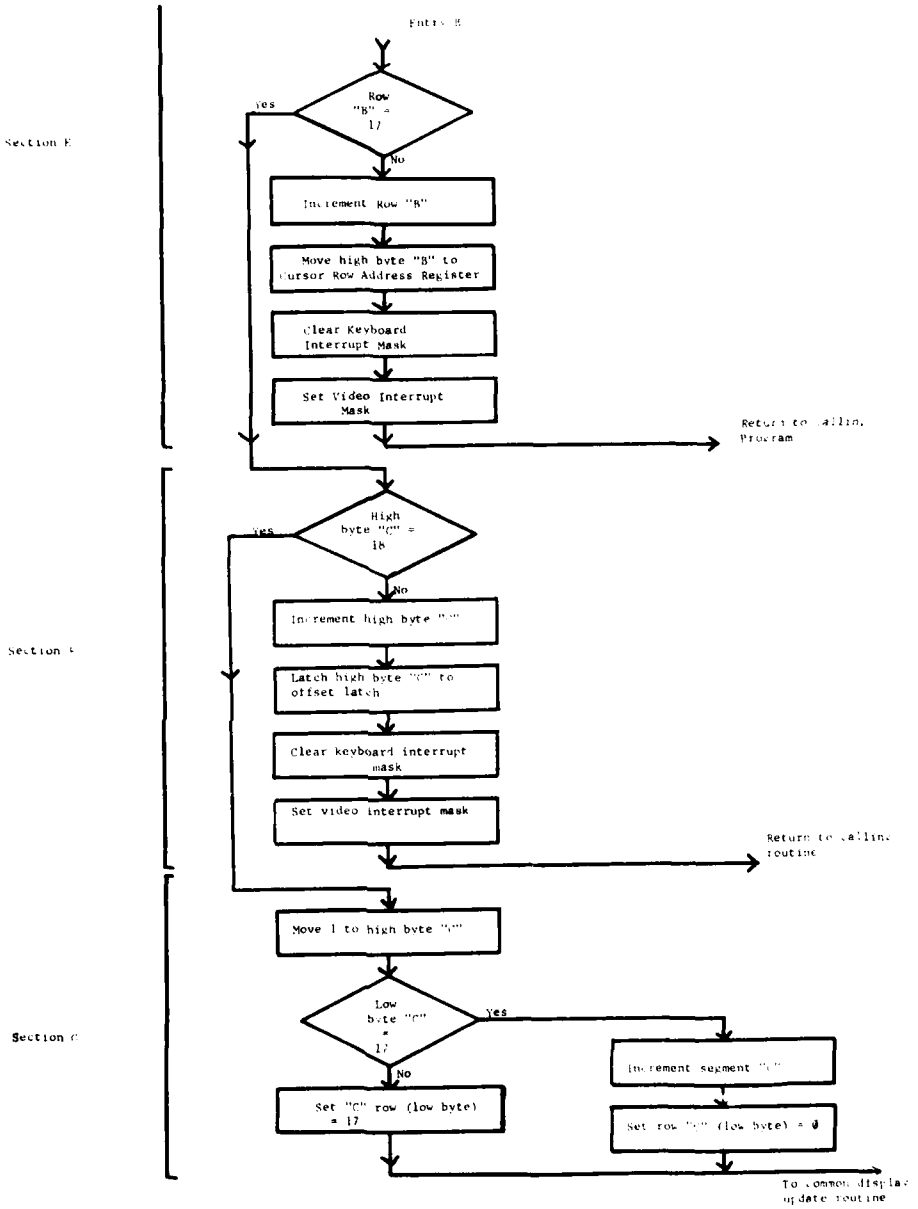


FIGURE F-2C CHARACTER WRITE ROUTINE, CONCLUDED

Character Deletion Routine

This routine deletes the character pointed to by the cursor. Video buffer memory is updated first, followed by the system memory, and the final character of the line which will be left blank.

Section A is used when the character to be deleted is the final character of a line.

Section B modifies the video buffer memory.

Section C modifies the system memory and returns control to the calling routine.

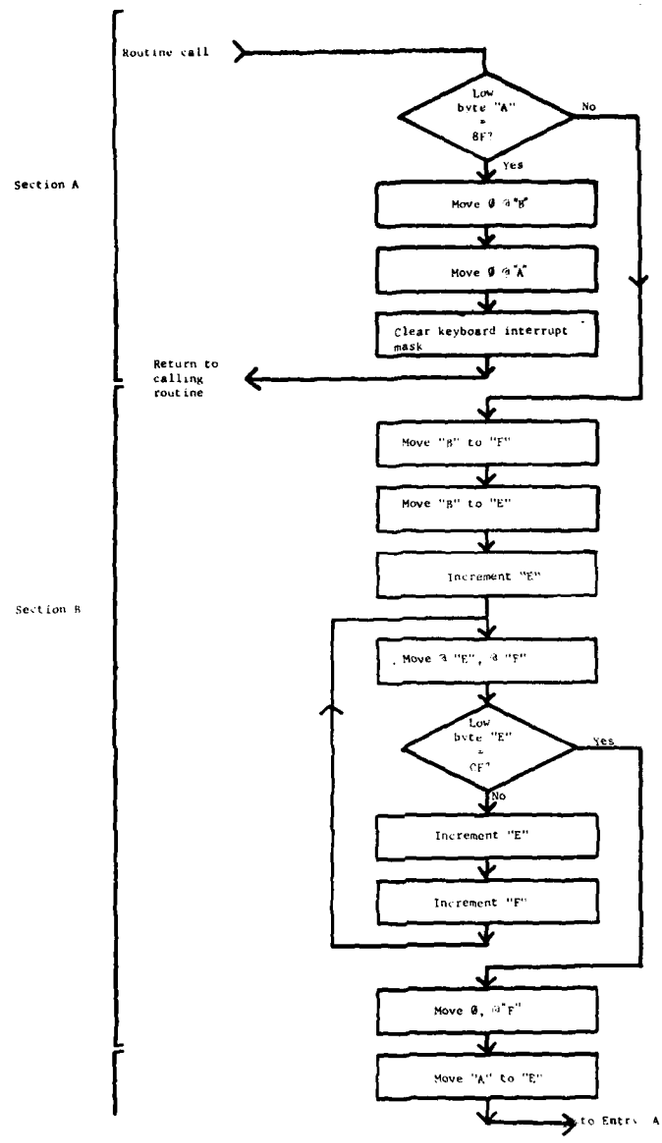


FIGURE F-3A CHARACTER DELETION ROUTINE

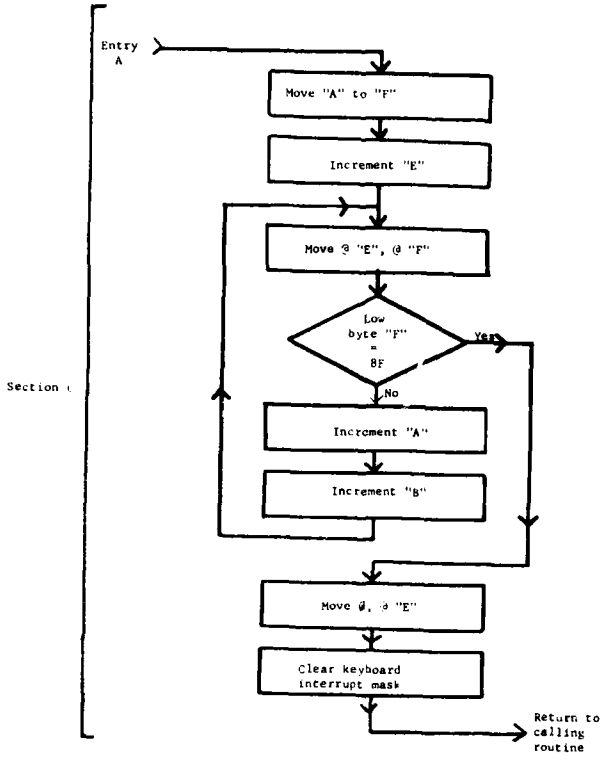
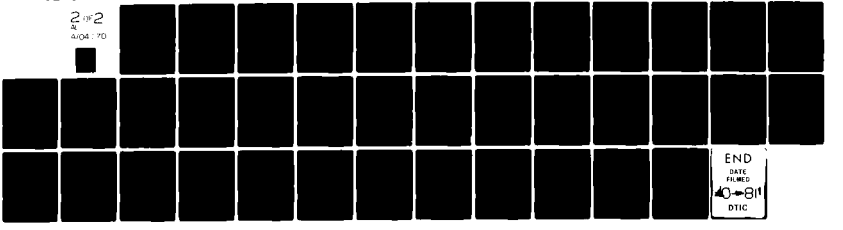


FIGURE F-3B CHARACTER DELETION ROUTINE, CONCLUDED

AD-A104 170 GENERAL ELECTRIC CO PITTSFIELD MA ORDNANCE SYSTEMS F/G 9/2
ELECTRICAL CHARACTERIZATION OF ADVANCED MICROPROCESSORS.(U)
JUN 81 B W HAJDUK, T M OSTROWSKI, B NEWTON F30602-80-C-0041
UNCLASSIFIED RADC-TR-81-126 NL



Character Insertion Routine

This routine inserts a blank character at the current cursor address. The final character of the line is lost.

Section A allows for the insertion of the final character of a line.

Section B modifies the video buffer memory.

Section C modifies the system memory and returns control to the calling routine.

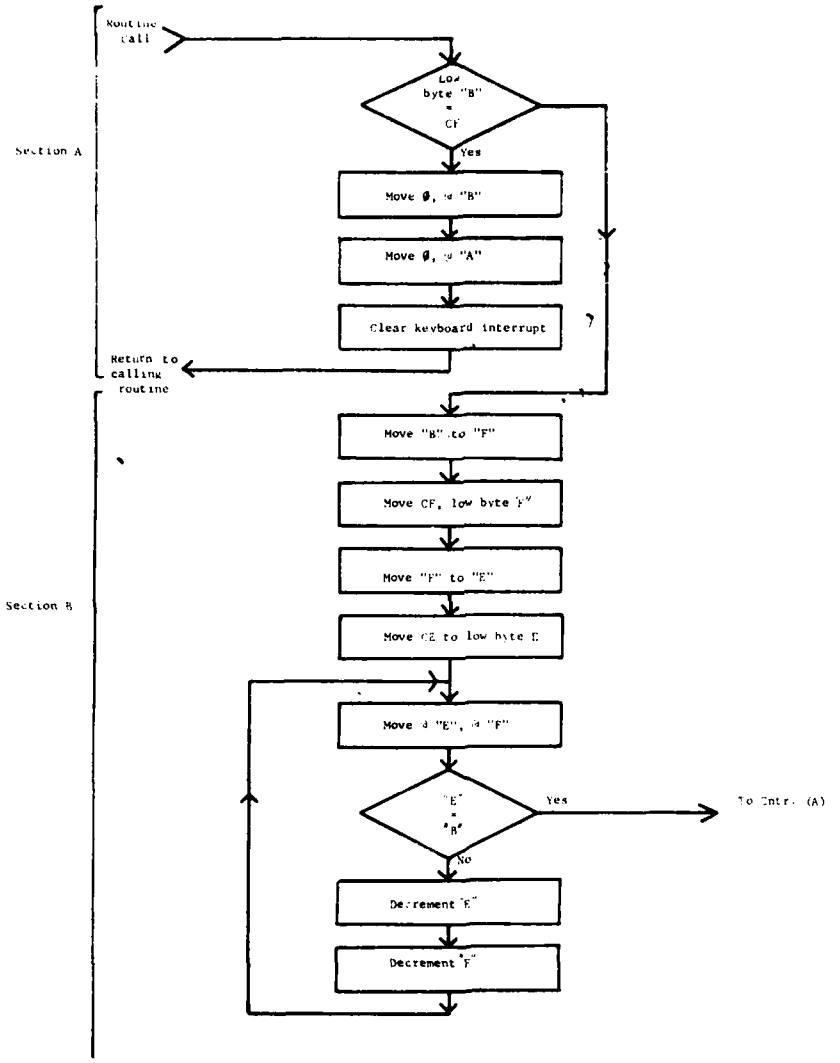


FIGURE F-4A CHARACTER INSERTION ROUTINE

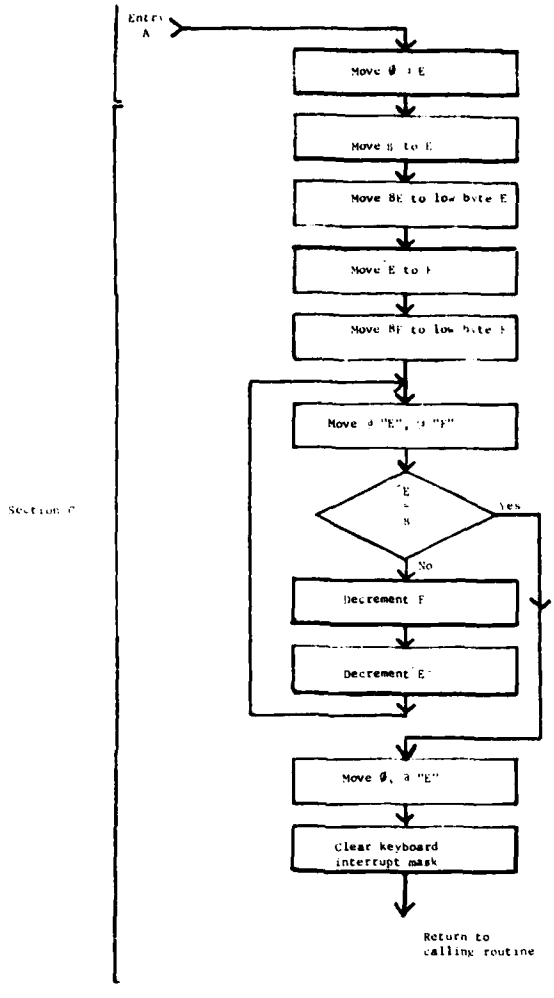


FIGURE F-4B CHARACTER INSERTION ROUTINE, CONCLUDED

Line Deletion Routine

This routine deletes a line from the system memory and decrements the maximum row count automatically. The video buffer memory is then updated using the display update routine.

Section A initializes the scratch pad registers.

Section B moves the characters.

Section C performs the update of pointers and sets the registers for the display update routine.

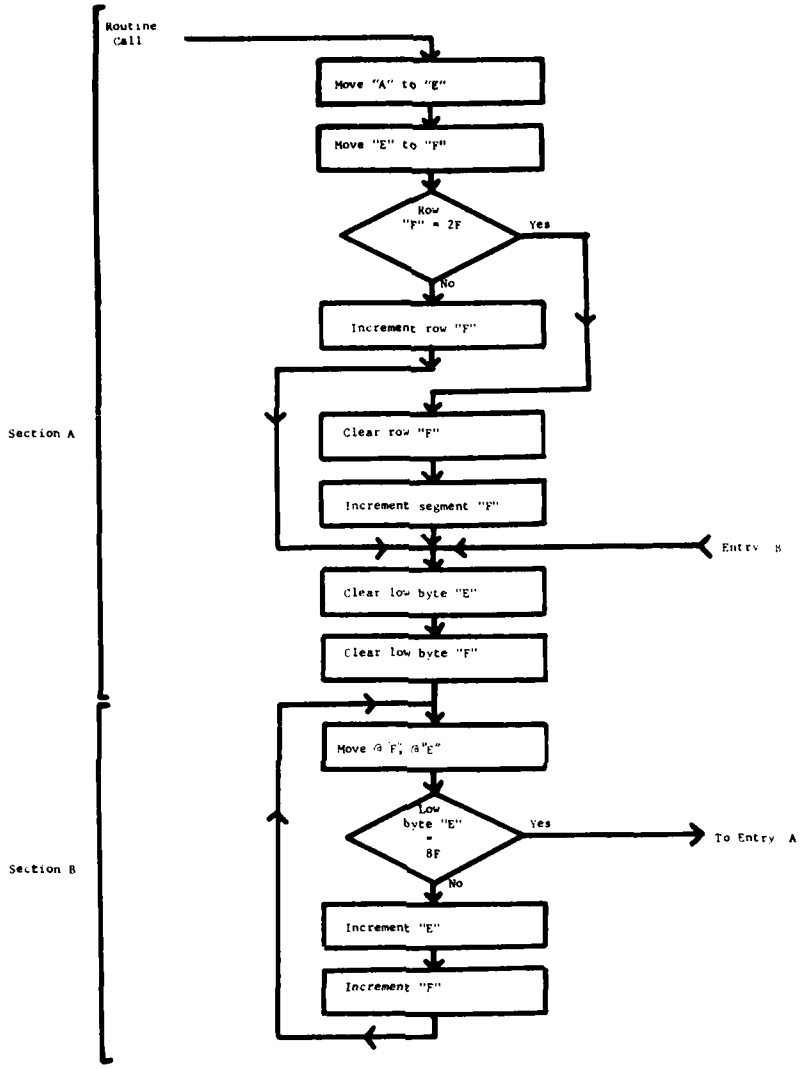


FIGURE F-5A LINE DELETION ROUTINE

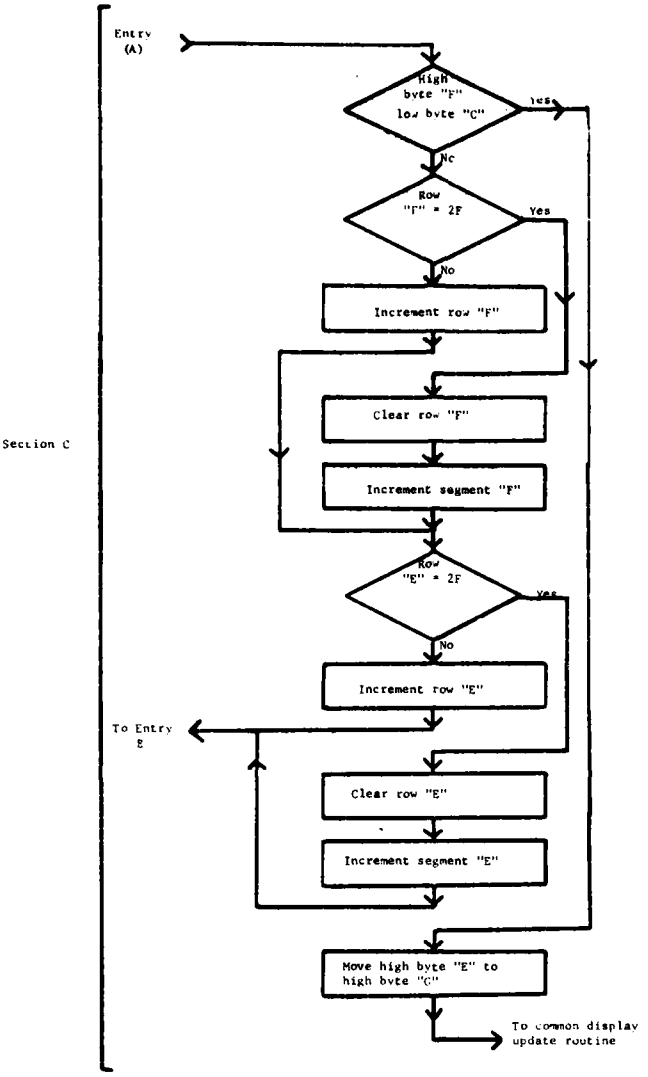


FIGURE F-5B LINE DELETION ROUTINE, CONCLUDED

Line Insertion Routine

This routine inserts a blank row at the cursor row address, automatically incrementing the system's maximum address count and using the common display update routine to modify the video buffer.

Section A initializes the routine's registers, updating the system maximum address with step 8.

Section B moves the characters in system memory and increments the pointers across the row.

Section C is a row decrement routine, decrementing the row address pointers back to the desired row.

Section D inserts the blanks into the addressed row and sets the parameters required for the use of the display update routine.

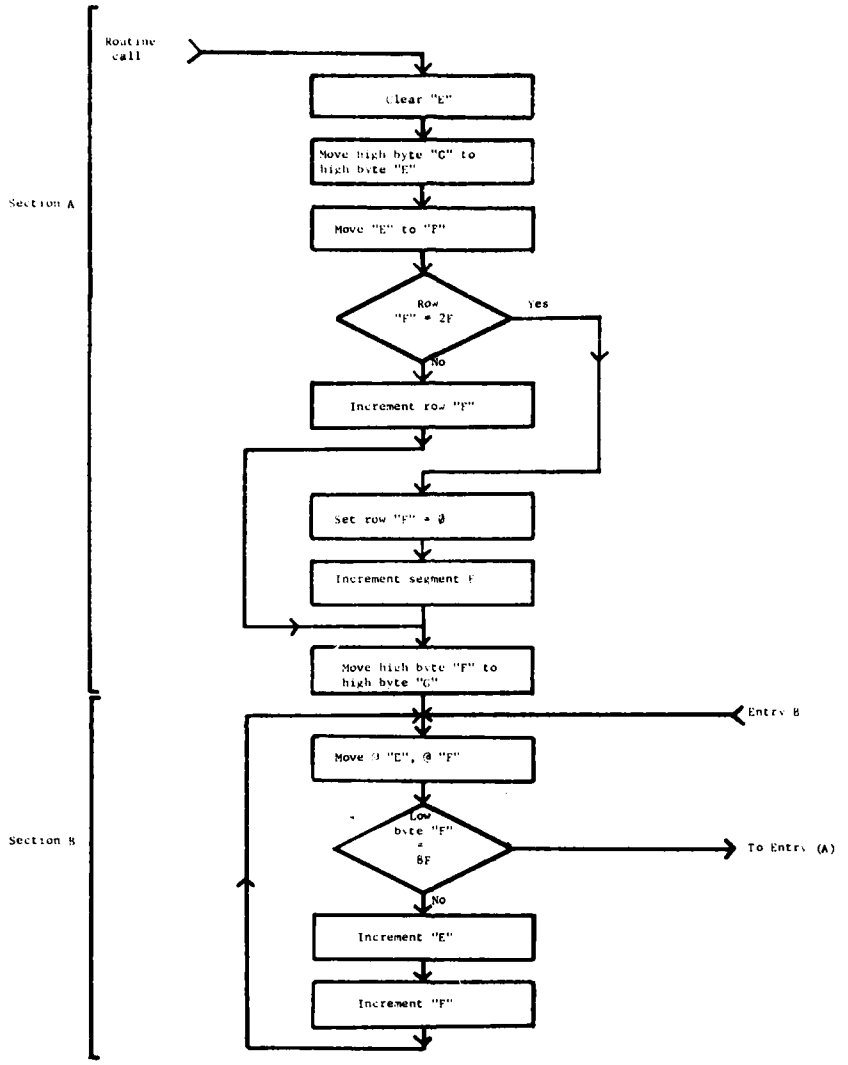


FIGURE F-6A LINE INSERTION ROUTINE

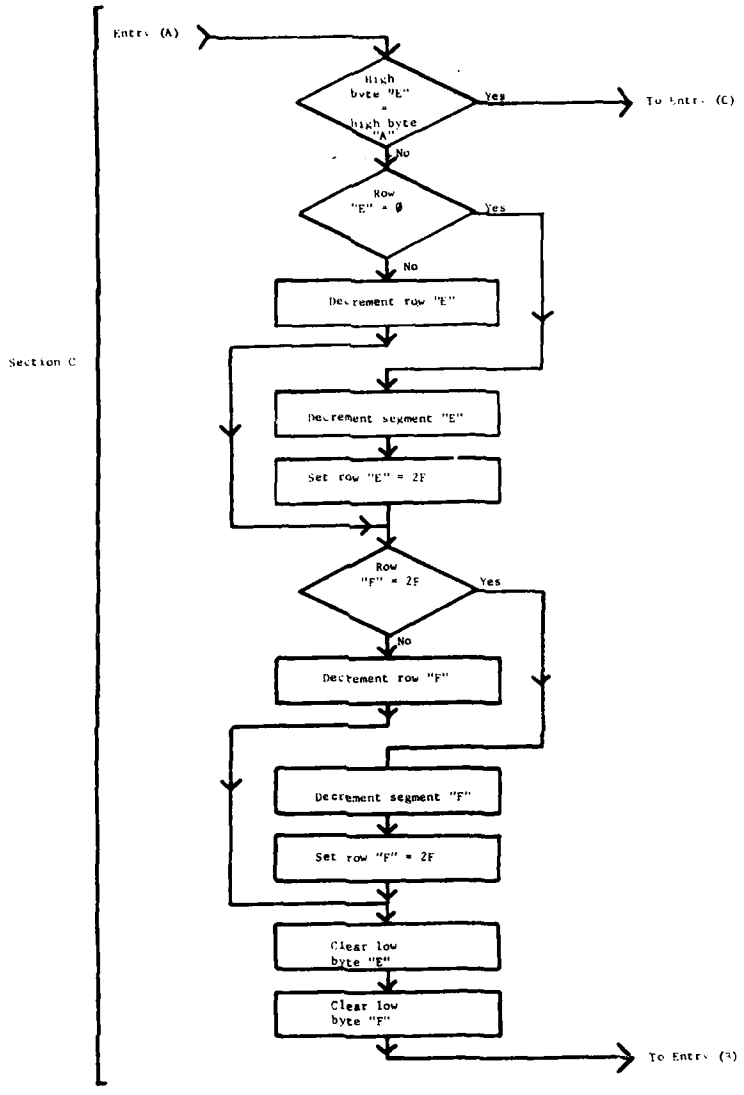


FIGURE F-6B LINE INSERTION ROUTINE, CONT'D

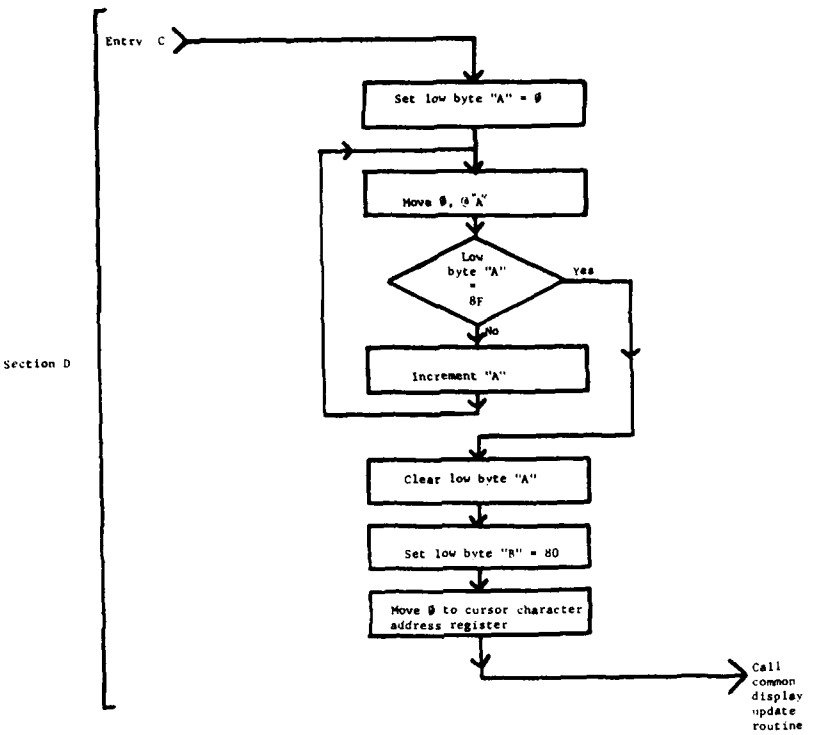


FIGURE F-6C LINE INSERTION ROUTINE, CONCLUDED

Left or Up Cursor Movement Routine

This routine allows for cursor movement to the left or upward (toward the initial line of the system memory) as indicated by the specific keyboard command. A cursor left command at the first character of a row will leave the cursor positioned at the last character of the row above. Automatic scrolling is allowed using this pair of commands, as the video buffer memory is updated by the display update routine.

Section A deals with the simple left movement of the cursor within a line.

Section B is an auto decrement section, its apparent complexity being the result of the x - y addressing desirable when using a VTAC. This section assumes no video buffer updating is required.

Section C assumes a video update is required, and decrements and initializes the registers required to use that routine.

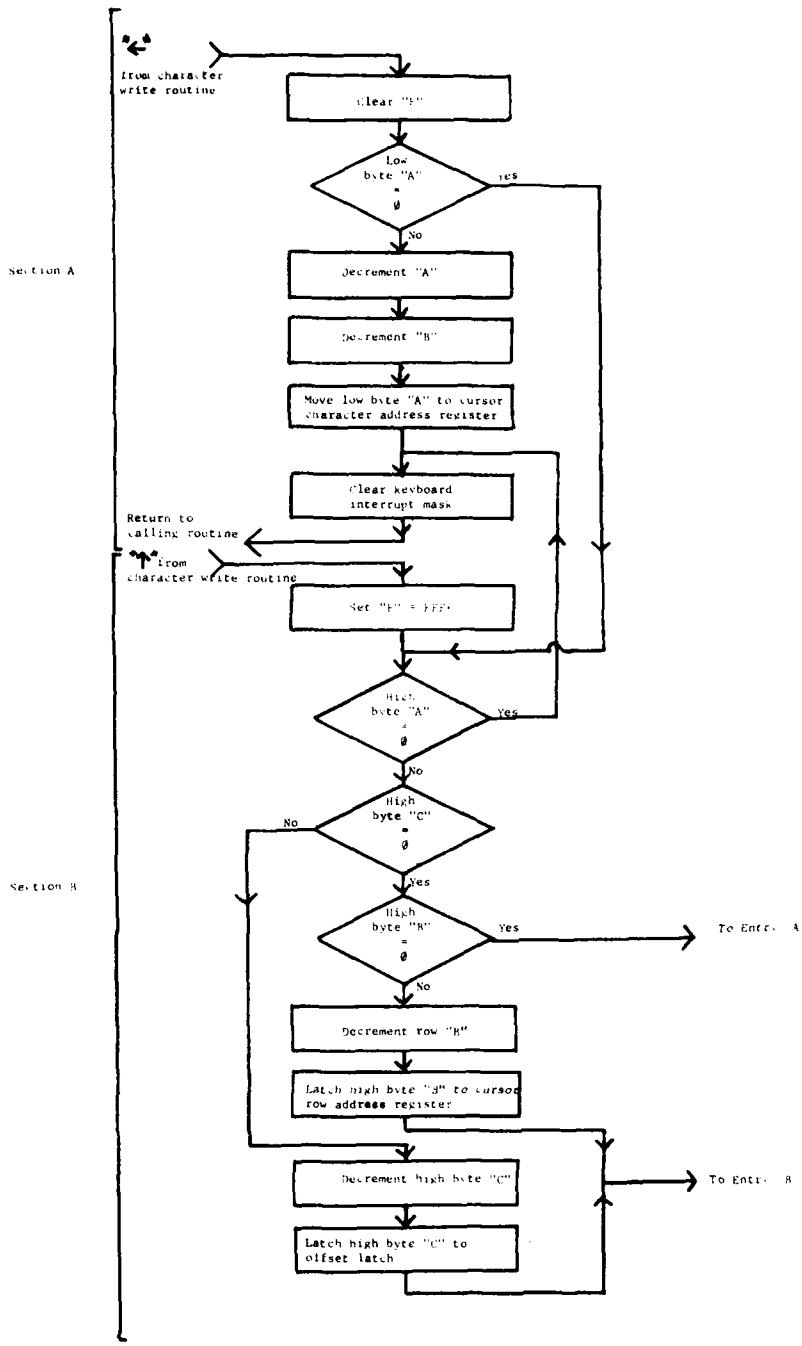


FIGURE F-7A LEFT OR UP CURSOR MOVEMENT ROUTINE

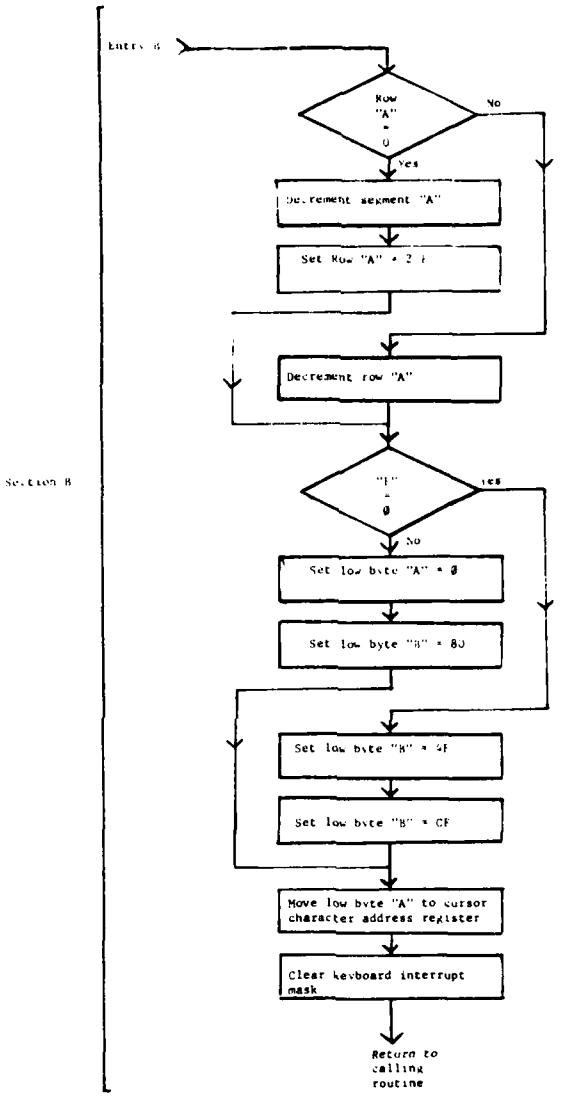


FIGURE F-7B LEFT OR UP CURSOR MOVEMENT ROUTINE, CONT'D

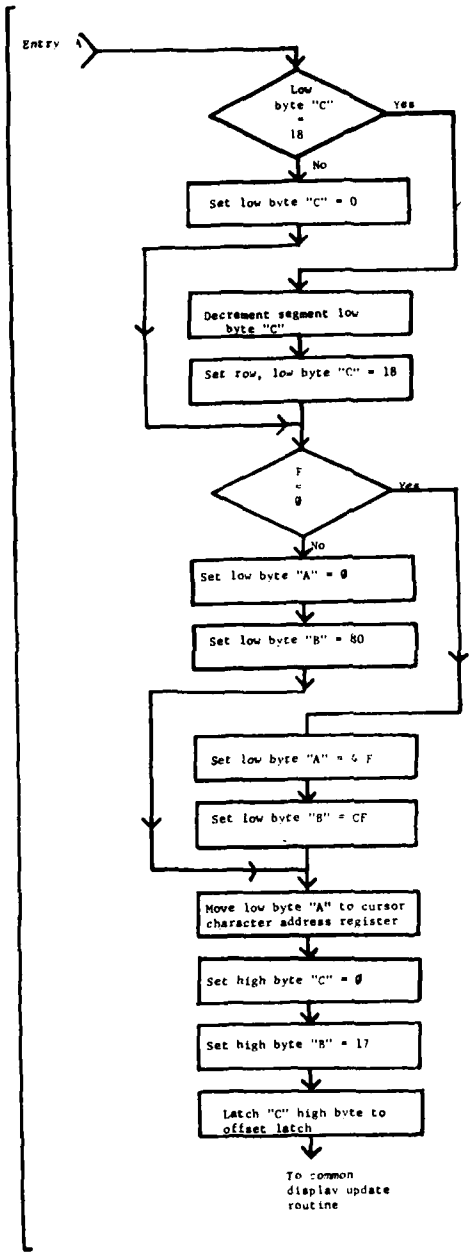


FIGURE F-7C LEFT OR UP CURSOR MOVEMENT ROUTINE, CONCLUDED

Right or Down Cursor Movement Routine

This routine moves the cursor to the right or downward (toward the final line of the system memory) on specific keyboard requests. Movement of the cursor beyond the last character of a row results in its being positioned at the first character of the next row. This routine allows automatic scrolling through the system memory, but will not allow movement of the cursor below the final printed line.

Section A provides the simple case of cursor movement within a line.

Section B determines if a display memory update will be necessary. If not, it adjusts the cursor position and returns control to the calling routine.

If a display update will be required, the routine proceeds to Section C, which sets the parameter registers and transfers control to the display update routine.

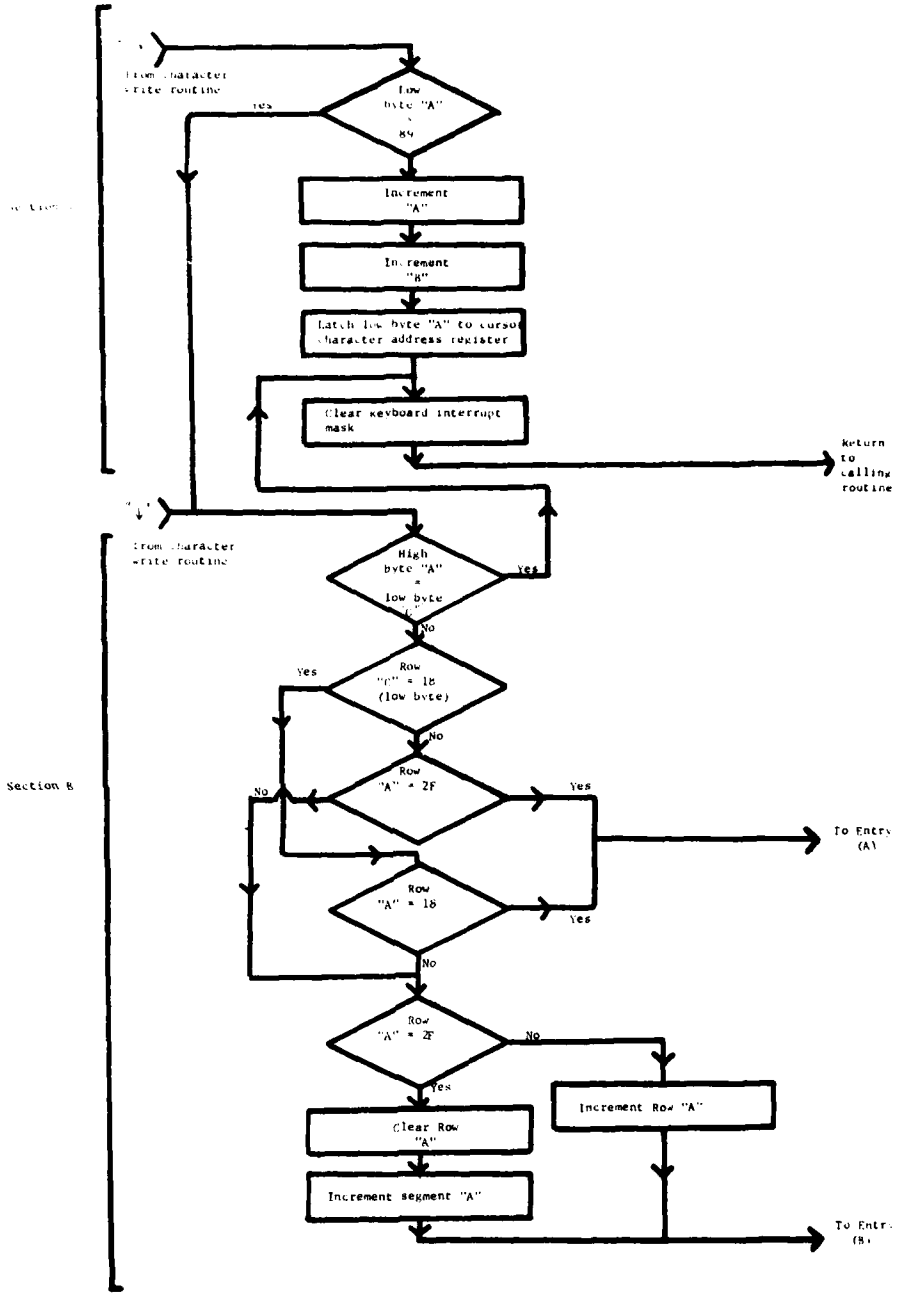


FIGURE F-8A RIGHT OR DOWN CURSOR MOVEMENT ROUTING

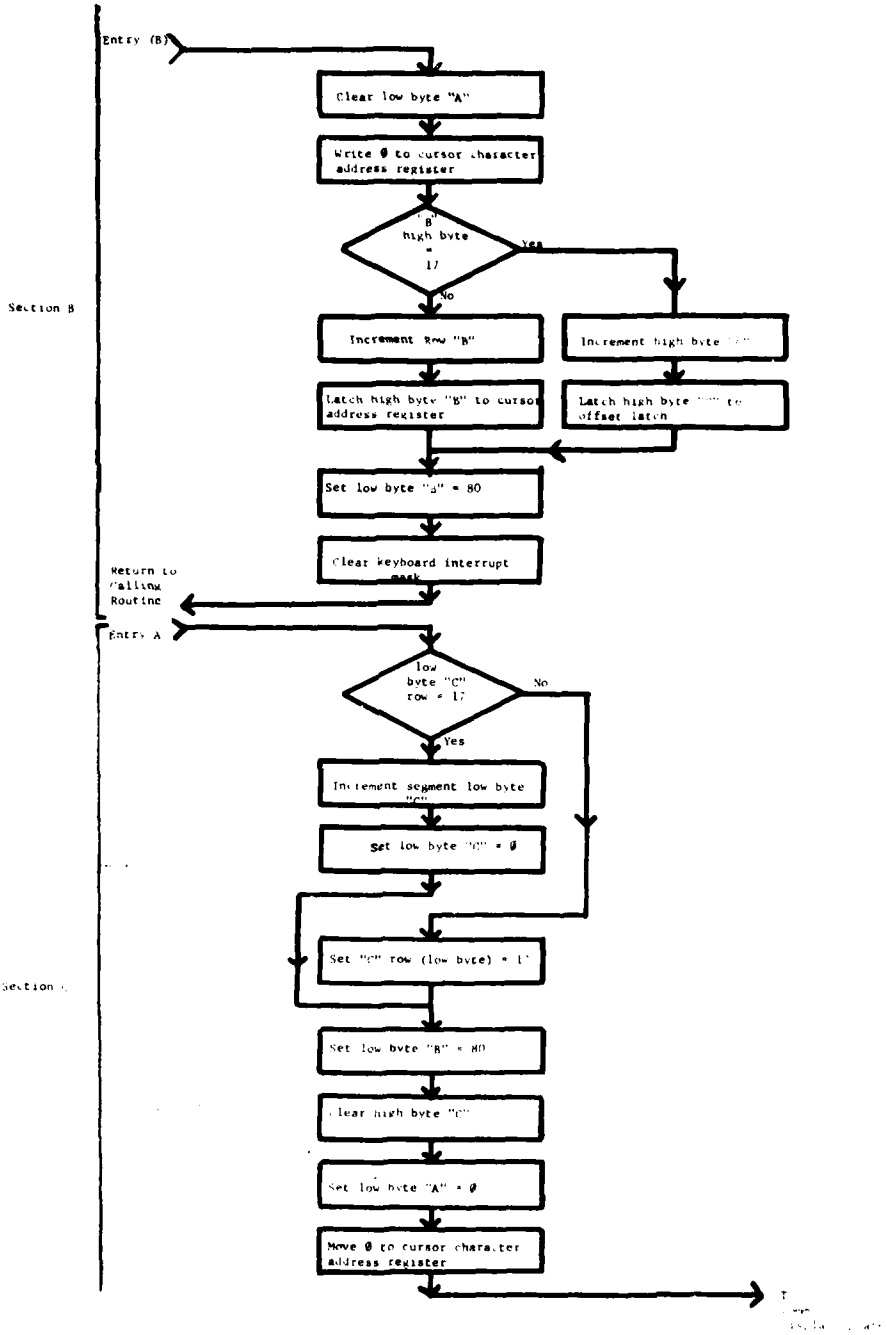


FIGURE F-8B RIGHT OR DOWN CURSOR MOVEMENT ROUTINE, CONCLUDED

Display Update Routine

This routine updates the video buffer memory by transferring data from the system memory. It cannot be called directly, but is automatically called by many routines (for instance, the line insertion/deletion routine).

No changes are made to the system memory by this routine.

Section A initializes the scratch pad registers used by the routine and sets the offset latch to zero.

Section B moves the data.

Section C increments the address pointers through both system and the video buffer memory, the latter, by the latching of offsets to the video buffer latch. On completion, interrupt masks are cleared and control returned to the calling routine.

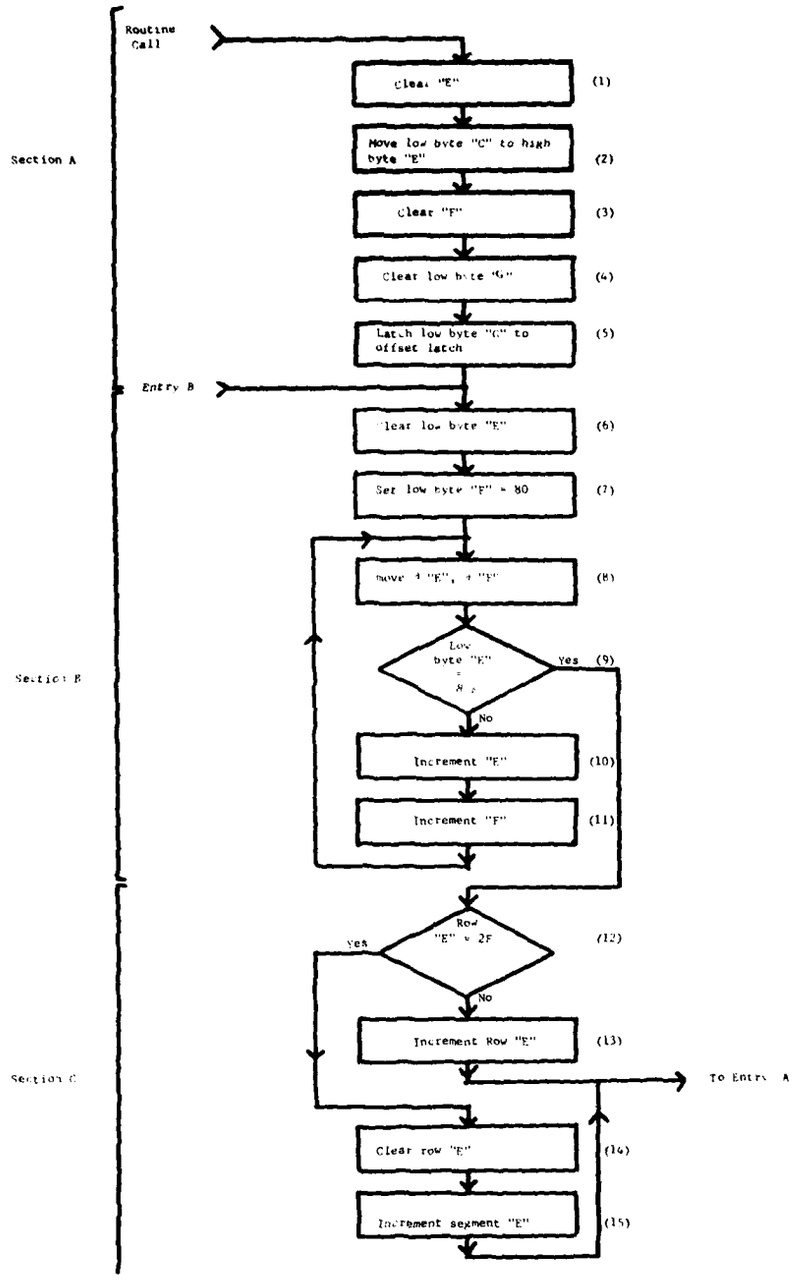


FIGURE F-9A COMMON DISPLAY UPDATE ROUTINE

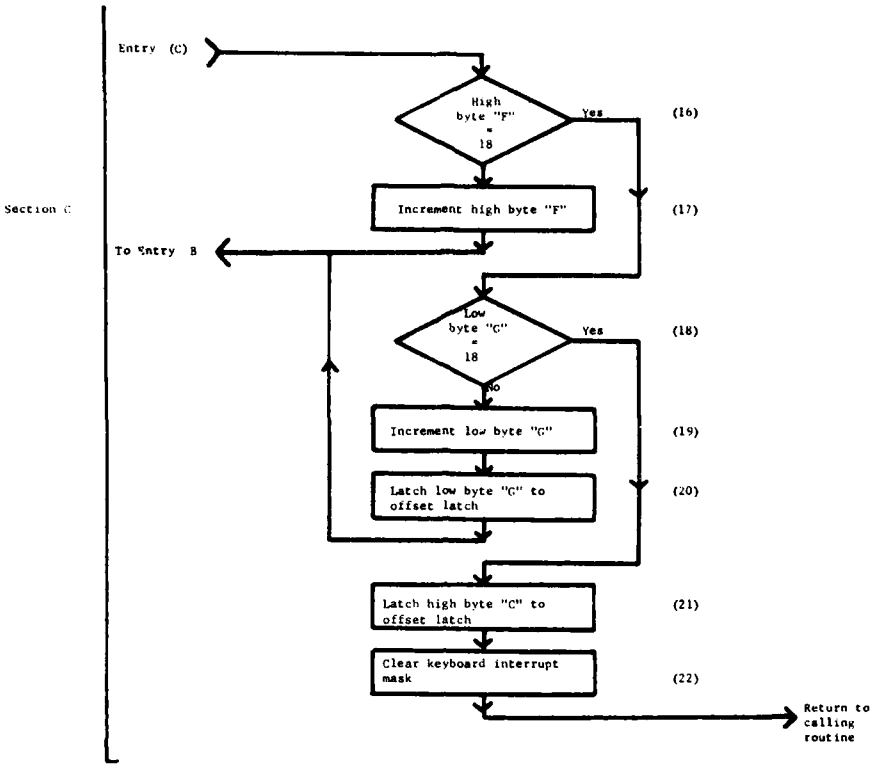


FIGURE F-9B COMMON DISPLAY UPDATE ROUTINE, CONCLUDED

Print Routine

This routine outputs the entire written system memory to the system's printing device. Neither system memory nor video buffer memory is modified by this routine.

Section A inhibits the video interrupt, allowing the output of data to run to completion, and outputs an entire line of characters before continuing to Section B.

Section B increments the row pointer through the system memory until the last written line is reached. At that point, interrupt masks are removed and control returned to the calling routine.

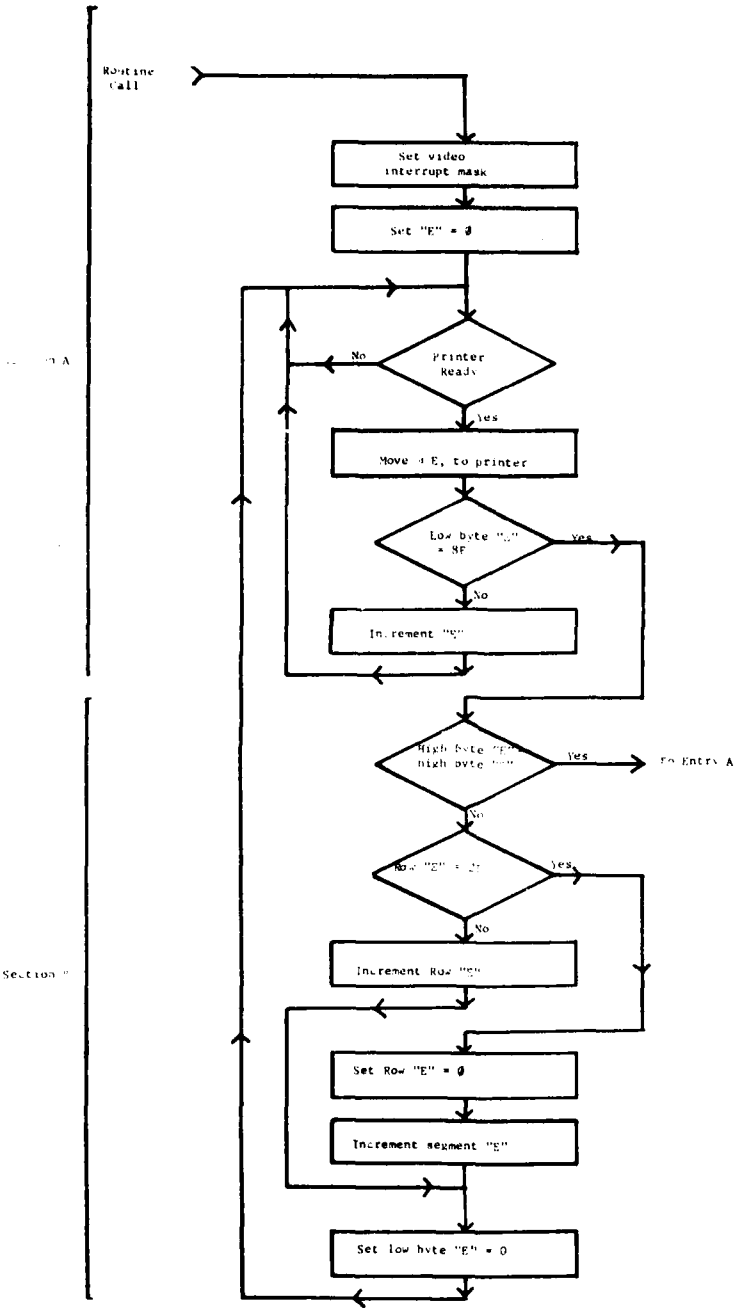


FIGURE F-10A PRINT ROUTINE

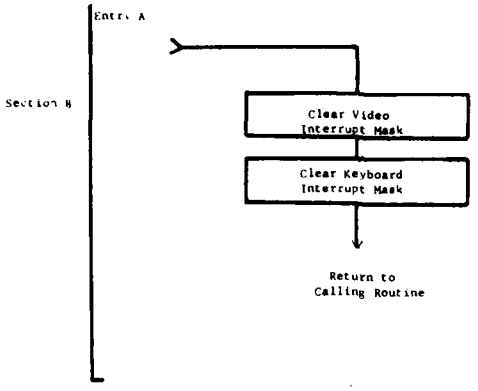


FIGURE F-10B PRINT ROUTINE, CONCLUDED

Final Mix Definition

The mix is determined by "running" the already defined task through the outlines routines as required. The results of this process are shown in Table F-1.

Note that this definition is free from bias in favor of any specific processor and is, due to the large number of instructions, insensitive to errors made in tabulating program steps. It should be noted that the task and programs required cannot be derived from the mix definition. Thus, most probable errors are compensated for and the pure statistical mix desired for benchmarking has been derived.

TABLE F-1 WORD PROCESSING INSTRUCTION MIX DEFINITION

	Command	Address Mode	
1.	Clear register (word)	N/A	742
2.	Clear register (byte)	N/A	730
3.	Set/Load byte	Immediate	679
4.	Set/Load word	Immediate	144
5.	Move byte, register to register	N/A	167
6.	Move word, register to register	N/A	1,157
7.	Move byte, memory to register	Indirect Register	107,520
8.	Move byte, register to memory	Indirect Register	119,040
9.	Move byte, register to memory (I/O)	Direct address	11,999
10.	Read memory to register, byte (I/O)	Direct address	12,384
11.	Clear memory (I/O)	Direct address	12,384
12.	Move # to memory, byte I/O	Direct address	148
13.	Move # to memory byte	Direct address	34,753
14.	Move # to memory byte	Indirect Register	1,233
15.	Increment register	N/A	203,744
16.	Increment byte register	N/A	24,445
17.	Decrement register	N/A	1,575
18.	Add register	Immediate	20
19.	Logical AND to register	Immediate	25
20.	Subtract, from register	Immediate	4
21.	Logical OR to register	Immediate	7
22.	Clear/Set bit	(I/O) Direct address	47,537
23.	Conditional jump taken	N/A	25,382
24.	Conditional jump, not taken	N/A	153,280
25.	Unconditional jump	N/A	98,344
		Total Commands	857,463

APPENDIX G

Z8001 TEST PROGRAMS

This appendix contains the programs that were developed and used during the characterization of the Z8001. The following describes the use of each program:

1. SHM02 .EDT: Z80, pages G-2 through G-10, is the test program used to characterize the Z8001. It uses a GO/NOGO functional test with worst case input timing conditions. The outputs are strobed at the vendor specified delays and pass/fail data is recorded as temperature, Vcc, frequency, duty cycle, and input levels are varied.

2. Z8000. PIN: Z80, pages G11 and G-12, is the Z8001 pin arrangement program used with the test program.

```
1.0100 * ****************************************************
1.0200 *
1.0300 *          TEKTRONIX S3270 TEST PROGRAM
1.0400 *
1.0500 *             CIRCUIT TEST ENGINEERING
1.0600 *          GENERAL ELECTRIC ORDNANCE SYSTEMS
1.0700 *             PITTSFIELD, MASSACHUSETTS
1.0800 *
1.0900 *
1.1000 * DUT PART OR DWG.#    :8001
1.1100 * DUT DESCRIPTION      :16BIT MICROPROCESSOR
1.1200 * DATE                 :10 JULY 1980
1.1300 * PROGRAMER            :
1.1400 *
1.1500 *
1.1600 *
1.1700 * FOR SUPPORTING DOCUMENTS,CONSULT ETEC AND
1.1800 * OTHER CARTNET FILES UNDER THE FOLLOWING
1.1900 * IDENTIFIED TEST SPECIFICATION NUMBER AND
1.2000 * ADAPTER NUMBERS, AS WELL AS LISTINGS OF THE
1.2100 * FOLLOWING IDENTIFIED DISK OR MAGNETIC TAPE
1.2200 * FILES.
1.2300 *
1.2400 *
1.2500 *
1.2600 * TEST SPECIFICATION  :DATA BOOK
1.2700 *
1.2800 * TEST TYPE/CONDITION :FUNCTIONAL,SHMOO PLOT
1.2900 *
1.3000 * OBJECT FILE         :8001
1.3100 *
1.3200 * EDIT FILE           :SHM02
1.3300 *
1.3400 * PIN ASSIGNMENT FILE :78000
1.3500 *
1.3600 * PATTERN FILE        :77900
1.3700 *
1.3800 * THIS LISTING IS     :TEKTEST EDIT
1.3900 *
1.4000 * SOCKET CARD         :#2060
1.4100 *
1.4200 * ****************************************************

3.0100 * PINLIST ASSIGNMENTS
3.0200 * ****************************************************
3.0300   PINLIST DATAT = TA00,TA01,TA02,TA03,TA04,TA05,TA06,TA
         07/
```

```
3.0000    TADR,TAD9,TAD10,TAD11,TAD12,TAD13,TAD14,TAD15
3.0500
3.0600    PINLIST INS = DATAI,STOP,MI,VI,NVI,SEGI,NMI/
3.0700    BUSRQ,WAIT,RESET,CLK
3.0750
3.0800    PINLIST DATAO = DAD0,DAD1,DAD2,DAD3,DAD4,DAD5,DAD6,DA
          D7/
3.0900    DADR,DAD9,DAD10,DAD11,DAD12,DAD13,DAD14,DAD15
3.1000
3.1100    PINLIST OUTS = DATAO,MO,MREQ,OSRI,ST0,ST1,ST2,ST3,SN0
          /
3.1200    SN1,SN2,SN3,SN4,SN5,SN6,BUSAK,RW,NS,BW,DS4,DSIO,AS
3.1300
3.1400    PINLIST ALL = DATAI,DATAO,STOP,MI,VI,NVI,SEGI/
3.1500    NMI,RESET,BUSRQ,WAIT,CLK,MO,MREQ,OSRI,ST0,ST1,ST2,ST3
          ,SN0/
3.1600    SN1,SN2,SN3,SN4,SN5,SN6,BUSAK,RW,NS,BW,DS4,DSIO,AS
3.1700
3.1800    PINLIST ASOSRI = AS,OSRI
3.1900    PINLIST ADSEG = DATAO,SN0,SN1,SN2,SN3,SN4,SN5,SN6

4.0100 * SUBROUTINES AND FUNCTIONS
4.0200 * ************************************************************
4.0300    SUBROUTINE HPWRS(V),HPWRCL(V,V),HPWRV(V,V,V):HP6129
4.0400    FUNCTION HPWRLS(V):HP6129
4.0500    SUBROUTINE FVS5(0),DVS5(0):MC3
4.0600    FUNCTION IVS5(0),MVS5(V):MC3
4.0700    SUBROUTINE SETEMP(V),RDTEMP(N):TP450A
4.0800    SUBROUTINE FVS7(0),DVS7(0):MC3
4.0900    FUNCTION MVS7(V),IVS7(0):MC3
4.1000    FUNCTION ATEMP(0):TP450A
4.1100    ARRAY TEMP2(8)
4.1200    ARRAY LDRV(11)
4.1300    ARRAY HDRV(11)
4.1400    ARRAY FREQ2(13)
4.1500    SUBROUTINE SAVTST(0):STATUS

5.0100 * DEVICE SPECIFICATION CONSTANTS
5.0200 * ************************************************************
5.0300    VCCNOM = 5.0V
5.0400    VCCMAX = 5.25V
5.0500    VCCMIN = 4.75V
5.0600    ICCMAX = 300MA
5.0700    VIH = 2.0V
5.0800    VIL = 0.8V
5.0900    VOH = 2.0V
5.1000    VOL = 0.8V
```

```
5.1100 * VCC LOOP CONSTANTS
5.1200   VMIN = 4.25V
5.1300   VMAX = 5.75V
5.1400   VINC = 0.25V
5.1500 * FREQUENCY LOOP CONSTANTS
5.1600   FMIN = 1E6
5.1700   FMAX = 6E6
5.1800   FINC = 1E6
5.1900 * DUTY CYCLE LOOP CONSTANTS
5.2000   DMIN = .2
5.2100   DMAX = .8
5.2200   DINC = .1
5.2700 * FLAG INITIALIZATION
5.2800   TEST = 0
5.2900   ABORT=0
5.3000   PRINT<12>CR,CR,"WHICH TEMPERATURE DO YOU WISH TO STAR
         T WITH?"
5.3100   PRINT<12>CR,"TYPE A 1 TO START WITH 125C."
5.3200   PRINT<12>CR,"TYPE A 2 TO START WITH 70C."
5.3300   PRINT<12>CR,"TYPE A 3 TO START WITH 25C."
5.3400   PRINT<12>CR,"TYPE A 4 TO START WITH 0C."
5.3500   PRINT<12>CR,"TYPE A 5 TO START WITH -55C."
5.3600   ACCEPT<12>CR,"ENTER YOUR CHOICE.     ",TTT,CR
5.3700   TESTTT EQ 116.01,6.06

6.0100 * LOG SERIAL NUMBER
6.0200 * ************************************************
6.0300   ACCEPT<12>CR,"ENTER DEVICE SERIAL NUMBER",SN,CR
6.0400   LOGMARKER<1>20
6.0500   LOGPARAMETRIC<1,9"SN">SN
6.0600   CONTINUE

9.0000 * POWER LOAD MODULE TO DUT
9.0100 * ********************************
9.0300   CALL 60.01
9.0400   GOTO(TTT)9.05,9.07,9.09,9.11,9.13
9.0500   LOOP 10.38 K=8,1,-1
9.0600   GOTO 10.0303
9.0700   LOOP 10.38 K=6,1,-1
9.0800   GOTO 10.0303
9.0900   LOOP 10.38 K=4,1,-1
9.1000   GOTO 10.0303
9.1100   LOOP 10.38 K=3,1,-1
9.1200   GOTO 10.0303
9.1300   LOOP 10.38 K=2,1,-1

10.0100 * TEMPERATURE CONTROL
```

```
10.0200 * ********************************************************
10.0303   IF(K EQ 2)10.38
10.0310   IF(K EQ 5)10.38
10.0313   IF(K EQ 7)10.38
10.0320   PRESET TEMP2=-55,-30,-1,25,0,70,100,125
10.0330   TEMP=TEMP2(K)
10.0333   PRINT CR,"**********************"
10.0400   PRINT CR,"TEMPERATURE=",TEMP;TO
10.0500   PRINT CR,"**********************",CR,CR
10.0600   LOGMARKER<1>30
10.0700   LOGPARAMETRIC<1,S"TEMP">K
10.0800   SETEMP(TEMP)
10.0900   *ATT 1S
10.1000   IF(ATTEMP EQ 0) 10.09
10.1100   *ATT 1S
10.1200   RDTEMP(TMP)
10.1300   IF(ABS(TEMP-TMP) LE 3.0) 10.18
10.1400   PRINT<12>"TEMPERATURE UNSTABLE",CR
10.1500   PRINT<12>"TEST ABORTED",CR,CR,CR
10.1550   ABORT=1
10.1600   GOTO 21.01
10.1700 * TEMP SOAK ROUTINE
10.1800   HP*RRS(1)
10.1900   PVSS
10.2000   HP*RCL(1,1A)
10.2100   HP*RRV(1,1,5V)
10.2200   WAIT 1S
10.2250   CONNECT LOAD1 ON OUT5
10.2300   V2=*VS5(5V)
10.2400   IF(4.99<V2<5.01) 10.29
10.2500   PRINT CR,"POWER SUPPLY ERROR DURING TEMP SOAK"
10.2600   PRINT CR,"TEST ABORTED"
10.2700   ABORT=1
10.2800   GOTO 21.01
10.2900   CYCLE=240NS
10.2910   PHASE 6=0NS FOR 120NS
10.2920   CONNECT TO PHASE ON CLK
10.2930   CONNECT INPUT TO DRIVER ON CLK
10.2940   HIDRIVE=5V ON CLK
10.2950   LODRIVE=0V ON CLK
10.2960   TOWIRTT CLK WITH ONE
10.2970   CONNECT INPUT TO DRIVER ON RESET
10.2980   HIDRIVE=3.5V ON RESET
10.2990   LODRIVE=0.4V ON RESET
10.3000   FORCE RESET WITH ZERO
10.3010   FORCE CLK WITH ONE;9Z
10.3020   BURST ON
```

```
10.3030    WAIT 400S
10.3040    BURST OFF
10.3050    INHIBIT RESET WITH ONE
10.3060    INHIBIT CLK WITH ONE
10.3500    DISCONNECT INPUT FROM DRIVER ON CLK
10.3550    DISCONNECT INPUT FROM DRIVER ON RESET
10.3600 *  CALL DRIVE LEVEL CONTROL
10.3700    CALL 11.01
10.3750    SAVTST
10.3800    CONTINUE
10.3900    GOTO 21.01

11.0100 *  DRIVE LEVEL CONTROL
11.0200 *  ************************
11.0300    LOOP 11.11 D=1,11
11.0400    PRESET LDRV=0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4,0.5,0.6,0.7,0
           .8
11.0500    PRESET HDRV=2.0,2.1,2.2,2.3,2.4,2.5,3.5,3.5,3.5,3.5,3.5,3
           .5
11.0600    PRINT CR,CR,"VTL = ",LDRV(D),"VOLTS        VTH = ",HDR
           V(D),"VOLTS"
11.0650    PRINT CR,"*********************************************"
11.0700    LOGMARKER<1>40
11.0800    LOGPARAMETRIC<1,S"DRV">D
11.0900 *  CALL VCC CONTROL
11.1000    CALL 13.01
11.1100    CONTINUE
11.1200    RETURN

13.0100 *  VCC CONTROL
13.0200 *  ************************************************
13.0300    LOOP 13.21 VCC=VMIN,VMAX,VINC
13.0400    PRINT CR,CR,"VCC= ",VCC," VOLTS    ,DUTY CYCLE= ",DMIN
           ," TO ",DMAX," BY ",DINC,CR
13.0500    LOGMARKER<1>50
13.0600    LOGPARAMETRIC<1,S"VCC">VCC
13.1000    HPWRRV(1,1,VCC)
13.1100    WAIT 1S
13.1110    IF(VCC EQ 5.0)13.115
13.1120    IF(VCC EQ 5.5)13.115
13.1130    IF(VCC EQ 6.0)13.115
13.1140    GOTO 13.12
13.1150    PS5=IVS5
13.1160    PRINT CR,"CURRENT = ",PS5," AMPS "
13.1200    V1=MVS5(VCC)
13.1300    IF(VCC-(VCC*.01)<V1<VCC+(VCC*.01)) 13.19
13.1400    PRINT CR,"POWER SUPPLY ERROR"
```

```
13.1500    PRINT CR,"TEST ABORTED"
13.1600    ABORT=1
13.1700    GOTO 21.01
13.1900 * CALL FREQUENCY CONTROL
13.2000    CALL 16.01
13.2100    CONTINUE
13.2200    RETURN

16.0100 * FREQUENCY CONTROL
16.0200 * ************************************************
16.0300    LOOP 16.09 F=1,13
16.0310    PRESET FREQ2=.25E6,.5E6,1E6,1.5E6,2E6,2.5E6,3E6,3.5E6
           ,4E6,4.5E6,5E6,5.5E6,6E6
16.0330    FREQ1=FREQ2(F)
16.0400    PRINT CR,"FREQ=",FREQ1," HZ     "
16.0450    FREQ=1/FREQ1
16.0460    FACTOR=FREQ/16N
16.0470    TRUNC1=INT(FACTOR)
16.0480    TRUNC2=TRUNC1*16N
16.0490    FREQ=TRUNC2
16.0500    LOGMARKER<1>60
16.0600    LOGPARAMETRIC<1,S"FREQ">FREQ1
16.0700 * CALL DUTY CYCLE CONTROL
16.0800    CALL 19.01
16.0900    CONTINUE
16.1000    RETURN

19.0100 * DUTY CYCLE AND TEST RESULTS CONTROL
19.0200 * ************************************************
19.0300 * THE DUTY CYCLE IS GIVEN IN FRACTIONS OF THE CYCLE
19.0400    LOOP 19.56 DCYCLE=DMIN,DMAX,DINC
19.0405    TEST=0
19.0410    CONNECT OUTPUT TO COMPARATOR ON OUTS
19.0420    MASK OUTS WITH ONE
19.0500    HICOMPARE = 2.4V ON ASDSRI
19.0600    LOCOMPARE = 0.4V ON ASDSRI
19.0800    PHASE 9 = 120NS FOR 10NS
19.1000    PHASE 10 = 80NS FOR 10NS
19.1100    COMPARE ASDSRI WITH PATTERN
19.1200    MASK ASDSRI WITH PATTERN
19.1300    GOTO 19.43
19.1400    HICOMPARE = 2.4V ON DSW
19.1500    LOCOMPARE = 0.4V ON DSW
19.1700    PHASE 9 = 0NS FOR 10NS
19.1800    COMPARE DSW WITH PATTERN
19.1900    MASK DSW WITH PATTERN
19.2000    GOTO 19.43
```

```
19.2100    HICOMPARE = 2.4V ON DSTO
19.2200    LOCOMPARE = 0.4V ON DSTO
19.2400    PHASE 9 = 0NS FOR 10NS
19.2500    COMPARE DSTO WITH PATTERN
19.2600    MASK DSTO WITH PATTERN
19.2700    GOTO 19.43
19.2800    HICOMPARE = 2.4V ON MREQ
19.2900    LOCOMPARE = 0.4V ON MREQ
19.3100    PHASE 9 = 0NS FOR 10NS
19.3200    COMPARE MREQ WITH PATTERN
19.3300    MASK MREQ WITH PATTERN
19.3400    GOTO 19.43
19.3500    HICOMPARE = 2.4V ON DATA0
19.3600    LOCOMPARE = 0.4V ON DATA0
19.3800    PHASE 9 = 100NS FOR 10NS
19.4000    PHASE 10 = 100NS FOR 10NS
19.4100    COMPARE DATA0 WITH PATTERN
19.4200    MASK DATA0 WITH PATTERN
19.4300    TEST = TEST+1
19.4400    CALL 50.01
19.4500    CALL 51.01
19.4600    CALL 52.01
19.4700    CALL 53.01
19.4800    IF(ERROR)19.51
19.4900    IF(TEST EQ 5) 19.51
19.5000    GOTO(TEST) 19.14,19.21,19.28,19.35
19.5100    PRINT NOT(ERROR):10,"  "
19.5300    LOGPARAMETRIC<1,S"DCYCLE">DCYCLE,NOT(ERROR)
19.5600    CONTINUE
19.5700    RETURN

21.0100 * TEST COMPLETION AND SHUTDOWN
21.0200 * ************************************************
21.0300    IF(ABORT)21.05
21.0400    PRINT CR,CR,"*****TEST COMPLETED*****",CR,CR
21.0500    VS7=0V AT 10MA
21.0600    DISCONNECT LOAD1 ON OUTS
21.0700    HP=PPS(1)
21.0750    SETEMP(25)
21.0800    STOP

50.0100 * PROGRAM INPUT TIMING
50.0200 * ************************************************
50.0300    CYCLE = FREQ
50.0400    PHASE 6 = 0NS FOR FREQ*DCYCLE
50.0500    CONNECT TO PHASE ON CLK
50.0600    PHASE 7 = FREQ-70NS FOR 70NS
```

```
50.0700    CONNECT TO PHASE ON SEGT
50.0800    PHASE 13 = FREQ*DCYCLE=70NS FOR 140NS
50.0900    PHASE 14 = FREQ*DCYCLE=70NS FOR 140NS
50.1000    CONNECT TO DATAPHASE ON DATAI
50.1100    RETURN

51.0100 * SET DRIVE LEVELS
51.0200 * ************************
51.0300    DISCONNECT INPUT FROM DRIVER ON INS
51.0400    HIDRIVE=HDRV(3) ON INS
51.0500    LODRIVE=LDRV(3) ON INS
51.0600    HIDRIVE=VCC-0.4 ON CLK
51.0700    LODRIVE=0.45 ON CLK
51.0800    INHIBIT INS WITH ONE
51.0900    CONNECT INPUT TO DRIVER ON INS
51.1000    RETURN

52.0100 * PROGRAM FORCE STATEMENTS
52.0200 * ************************************************
52.0300    FORCE INS WITH PATTERN
52.0400    FORCE CLK WITH PATTERN;RZ
52.0500    FORCE SEGT WITH PATTERN;RZ,INVERT
52.0600    FORCE DATAI WITH PATTERN;RZ
52.0700    INHIBIT DATAI WITH PATTERN
52.0800    RETURN

53.0100 * THE MOVE ROUTINE
53.0200 * ************************************************
53.0300    LOAD FROM CORE 77900 TO ALL WITH FI,CM
53.0400    TRIGGER 1
53.0500    MOVE REGISTER(516) TO ALL WITH FI,CM
53.0600    INHIBIT INS WITH ONE
53.0650    MASK OUTS WITH ONE
53.0700    RETURN

60.0000 * SETUP 2.5V LOAD MODULE SUPPLY
60.0100 * ************************************
60.0200    FVS7
60.0300    VS7=2.5V AT 100MA
60.0400    WAIT 50MS
60.0500    V7=MVS7(5)
60.0600    I7=IVS7
60.0700    IF(2.49<V7<2.51)60.12
60.0800    PRINT CR,"VOLTAGE ERROR ON LOAD MODULE SUPPLY"
60.0900    PRINT CR,"TEST ABORTED"
60.1000    ABORT=1
60.1100    GOTO 21.01
```

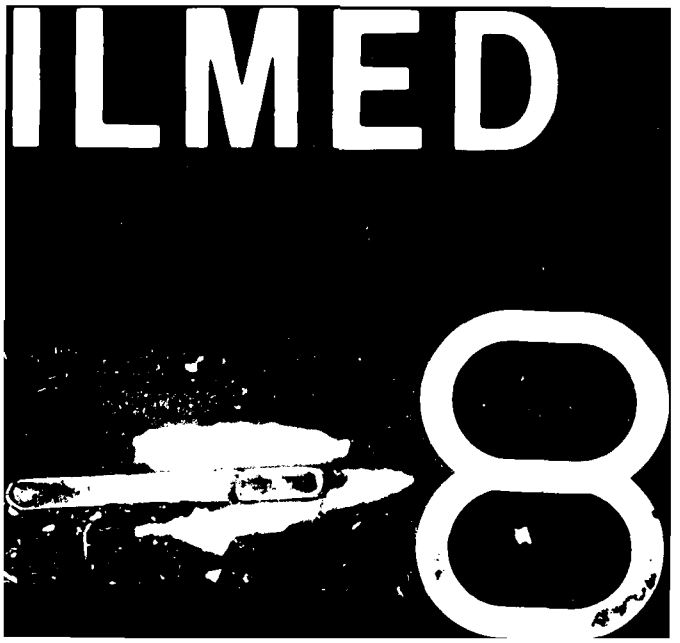
```
60.1200    IF(T7 LT 100)&)60.17
60.1300    PRINT LP,"CURRENT ERROR ON LOAD MODULE SUPPLY"
60.1400    PRINT CR,"TEST ABORTED"
60.1500    ABORT=1
60.1600    GOTO 21.01
60.1700    RETURN
```

LINE NUMBER	SECTOR NUMBER	PIN NAME	DUT PIN OR COMMENT
1.0000	1XAI	IAD0	ADDR/DATA INPUT
2.0000	44ZAI	IAD1	ADDR/DATA INPUT
3.0000	46XAI	IAD2	ADDR/DATA INPUT
4.0000	48ZAI	IAD3	ADDR/DATA INPUT
5.0000	54XAI	IAD4	ADDR/DATA INPUT
6.0000	50XAI	IAD5	ADDR/DATA INPUT
7.0000	56ZAI	IAD6	ADDR/DATA INPUT
8.0000	58XAI	IAD7	ADDR/DATA INPUT
9.0000	62XAI	IAD8	ADDR/DATA INPUT
10.0000	3YAI	IAD9	ADDR/DATA INPUT
11.0000	5WAI	IAD10	ADDR/DATA INPUT
12.0000	9WAI	IAD11	ADDR/DATA INPUT
13.0000	11YAI	IAD12	ADDR/DATA INPUT
14.0000	13WAI	IAD13	ADDR/DATA INPUT
15.0000	18XAI	IAD14	ADDR/DATA INPUT
16.0000	15YAI	IAD15	ADDR/DATA INPUT
17.0000	64ZAO	OAD0	ADDR/DATA OUTPUT
18.0000	43YAO	OAD1	ADDR/DATA OUTPUT
19.0000	45WAO	OAD2	ADDR/DATA OUTPUT
20.0000	47YAO	OAD3	ADDR/DATA OUTPUT
21.0000	53WAO	OAD4	ADDR/DATA OUTPUT
22.0000	49WAO	OAD5	ADDR/DATA OUTPUT
23.0000	55YAO	OAD6	ADDR/DATA OUTPUT
24.0000	57WAO	OAD7	ADDR/DATA OUTPUT
25.0000	61WAO	OAD8	ADDR/DATA OUTPUT
26.0000	2XAO	OAD9	ADDR/DATA OUTPUT
27.0000	4ZAO	OAD10	ADDR/DATA OUTPUT
28.0000	6XAO	OAD11	ADDR/DATA OUTPUT
29.0000	10XAO	OAD12	ADDR/DATA OUTPUT
30.0000	12ZAO	OAD13	ADDR/DATA OUTPUT
31.0000	17XAO	OAD14	ADDR/DATA OUTPUT
32.0000	14XAO	OAD15	ADDR/DATA OUTPUT
33.0000	63YAI	STOP	STOP
34.0000	51YAI	MI	MICRO-IN
35.0000	7YAI	VI	VECTORED INTERRUPT
36.0000	8ZAI	NVI	NON-VECTORED INTERRUPT
37.0000	16ZAI	SEGT	SEGMENT TRAP
38.0000	19YAI	NMI	NON-MASKABLE INTERRUPT
39.0000	20ZAI	RESET	RESET
40.0000	32ZAI	BUSRQ	BUS REQUEST
41.0000	35YAI	WAIT	WAIT
42.0000	41WAI	CLK	CLOCK
43.0000	22XAO	MO	MICRO-OUT
44.0000	23YAO	MREQ	MEMORY REQUEST
45.0000	24ZAO	DSRI	DATA STROBE READ

46.0000	28X40	ST0	STATUS
47.0000	27X40	ST1	STATUS
48.0000	26X40	ST2	STATUS
49.0000	25X40	ST3	STATUS
50.0000	34X40	SN0	SEGMENT NUMBER
51.0000	33X40	SN1	SEGMENT NUMBER
52.0000	32X40	SN2	SEGMENT NUMBER
53.0000	29X40	SN3	SEGMENT NUMBER
54.0000	52Y40	SN4	SEGMENT NUMBER
55.0000	59Y40	SN5	SEGMENT NUMBER
56.0000	60Y40	SN6	SEGMENT NUMBER
57.0000	36Y40	BUSAK	BUS ACKNOWLEDGE
58.0000	37X40	RW	READ/WRITE
59.0000	38X40	NS	NORMAL/SYSTEM MODE
60.0000	39Y40	BW	BYTE/WORD
61.0000	30X40	VCC	SUPPLY VOLTAGE
62.0000	21X40	DSW	DATA STROBE WRITE
63.0000	31Y40	DSIO	DATA STROBE INPUT/OUTPUT
64.0000	40Y40	AS	ADDRESS STROBE

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


ILMED
8